B. Comp Dissertation Final Report

"Online Judge" for data structures and algorithms course

By

Ivan Reinaldo

Department of Computer Science

School of Computing

National University of Singapore

2013/2014

B. Comp Dissertation Final Report

"Online Judge" for data structures and algorithms course

By
Ivan Reinaldo

Department of Computer Science
School of Computing
National University of Singapore
2013/2014

Project No: H160080
Advisor: Dr. Steven Halim
Deliverables:

- Report: 1 volume

Abstract

This project is a data-structure learning project which is part of the project on unified and interactive web-based visualization of various classical and non-classical algorithms at http://www.comp.nus.edu.sg/~stevenha/visualization. This project aims to build an interactive learning website on top of the visualizations for student to refine and verify their knowledge on data structures and algorithms. This report will include changes made in software system design, some of the visualizations implemented in the data-structure learning project, and the online judge system to test basic understanding of the students.

This project is built mainly on HTML5, JavaScript, and CSS3, with some PHP and MySQL for online judge functionalities and using D3.js library for a more interactive visualization. Link to temporary project website: http://algorithmics.comp.nus.edu.sg/~onlinequiz/staging/. Link to temporary repository of the project source code: https://github.com/stevenhalim/visualgo.

Subject descriptors:

Applied computing → E-learning

Applied computing → Computer-assisted instruction

Human-centered computing → Graph drawings

Social and professional topics → Computer science education

Software and its engineering → Object oriented architectures

Computer systems organization → n-tier architectures

Software and its engineering → Abstraction, modeling and modularity

Software and its engineering → Software usability

Keywords:

eLearning, Data Structure and Algorithms, Computer Animation

Implementation software:

HTML5, JavaScript 1.7, CSS3, PHP 5.3.3, MySQL

Acknowledgement

I am thankful to the National University of Singapore for providing me a chance to learn and take this project, and especially to Dr. Steven Halim for being a wonderful supervisor who not only guided me, but also took part in coding the project. I also thank my teammate Rose Marie Tan for her various contributions, especially for the beautiful UI design which made the visualization website more attractive, and my teammate Duy Nguyen for his various contributions, especially on his implementation of graph drawing functionalities which are very challenging. Last but not least, I also want to thank my previous employer A*STAR Business Analytics Translational Centre (BATC) for introducing D3.js to me and taught me numerous software engineering techniques which I applied in this project.

Table of Content

1 Introduction

1.1 About the Project

This report describes my contribution on a long-term project called "VisuAlgo". VisuAlgo is a project started by Dr. Steven Halim since 2011, with the aim of providing an interactive learning platform on Data Structures and Algorithms topics. In order to achieve these, currently VisuAlgo supports two main functionalities, which are teaching students about Data Structures and Algorithms (henceforth referred to as "visualization tool"), and assessing students' understanding on Data Structures and Algorithms (henceforth referred to as "online judge tool").

During this academic year, the online judge tool is the focus of the project. This is mainly because the visualization tool has been implemented extensively in the previous academic years, and had supported a large amount of Data Structures and Algorithms visualization. My main contribution is to build the online judge tool from scratch and integrate them with the existing visualization tool. However, instead of integrating the online judge tool with the existing visualization tool, I have to build a new visualization tool replacing the existing visualization tool due to technical reasons outlined in this report. The final product consists of a website and a server-side component with the following features (my part is shown in *italic and underline*):

Visualization Tool

- Visualisations using the new User Interface (UI) and *Scalable Vector Graphics (SVG) animation*:
  Binary Search Tree/AVL Tree, Binary Heap, Union Find, Bitmask, *Minimum Spanning Tree*, Single-Source Shortest Paths, Suffix Tree, Suffix Array, and Geometry, *as well as a system to display the animation*
- Visualisations not yet updated to use the new interface and still using HTML5 Canvas:
  Sorting, Linked List, Stack, Queue, Recursion, Graphs, Graph Traversal, Segment Tree, Binary Indexed Tree, Max Flow, and Graph Matching

Online Judge Tool

- Client-side

  A Training page for students' self-practice, a Test page for formal regulated tests, an Answer Key page for students to review their answers after the test, a Scoreboard of test results, and an Admin Control Panel for the examiner to control, customise and monitor the test in progress

- *Server-side*

  Question generators and verifiers on the following topics: *Binary Search Tree/AVL Tree*, *Binary Heap*, *Union Find*, Bitmask, Minimum Spanning Tree, Single-Source Shortest Paths, Graph Traversal, Graph Data Structures, *as well as all other infrastructures and database*

1.2 Motivation

The motivation of this project was an effort to improve the teaching methods on Data Structures and Algorithms which are usually done in the following ways (Halim et al, 2010):

- Using pre-scripted PowerPoint presentation (or other presentation applications) with diagrams / animations of the algorithms
- Using hand-drawn diagrams of the algorithms on the board / writing on projected paper / writing on monitor using stylus pen
- Providing links to pre-existing algorithms visualizations

As indicated in Dr. Halim's paper, this project aims to provide a better learning method than the methods mentioned above, by providing a unified platform for visualization of various Data Structures and Algorithms.

The motivation of the online judge tool stems from Dr. Halim's experience in conducting graded assessments. He found that there are questions which are repetitive and trivial in nature, but is tedious to create and verify. Thus, he developed the concept of automating the process of creating and verifying such questions, which becomes the online judge tool.

1.3 Project Objectives

Improving Learning Experience for Students

During the previous academic years, the project has gathered positive responses from the students, as seen in this module survey during AY 2012/2013:

Question	CS3233 (24 responses)	CS2020 (7 responses)
Does the visualization help you understand the algorithm taught better?	Yes (13/54.1%) No (0/0.0%) Neutral (11/45.8%)	Yes (5/71.4%) No (0/0.0%) Neutral (2/28.5%)
When do you use the visualization? (select all that applicable to you)	In class (8/33.3%) After class (4/16.6%) Before test (14/58.3%) Others (6/25.0%)	In class (1/14.2%) After class (0/0.0%) Before test (5/71.4%) Others (3/42.9%)

There are two main points that we can observe in this data: students thinks that visualization tools are good addition to the learning process, and that most students only use the visualization tools before the test. Regarding the first point, it is important to note that no students think that the visualization tool is detrimental to the learning process, and that the number of students who benefit from this tool is more than 50%. This is consistent with Fleming's VARK model, which states that 60-65% people are visual learners and thus learn best by using pictures and visual aids (Fleming, 2006). This means that the visualization tool do help students in their learning process. This result is intuitive, because with the visualization tool students can learn the materials using their own data set and receive confirmation on their understanding, instead of having to rely on hardcoded examples. In addition, according to Dr. Halim, students found the visualization page impressive when they see it for the first time. Visualization tool acts as a virtual lecturer that explains how Data Structures and Algorithms work. If

students are still unclear about the topic, they will have an assurance that there is always a virtual lecturer that will help them to re-explain the topic

In the second point, it is shown that students mostly use the visualization tool before tests, and stopped using the tool after the test. This means that the visualization tool previously functions as a review material for students, and thus is open to improvements. In particular, the tools in this project should be used more often when there is no graded test. To achieve this, one improvement that the project brings this academic year is the online judge tool. This tool will allow students to immediately confirm their understanding on the topics after each lectures / tutorials by enabling them to practice on pseudo-randomly generated questions. This tool will also greatly help as review material for tests, because now students can practice on questions they have never seen before. Online judge tool acts as a virtual examiner that allows students to practice on new questions.

Ultimately, this project will improve learning experience for students, by providing students with the means to learn the topics and practice on questions on their own.

Enhancing Teaching Experience for Teachers

As mentioned in motivation section, teaching Data Structures and Algorithms are usually done by either providing pre-scripted PowerPoint examples, hand-drawn diagrams, or providing links to existing visualization tools. There are problems in each of the approach.

Pre-scripted PowerPoint examples are the most common method chosen, because they do not present any difficulties for teacher during lectures. However, there are several problems associated with them, some of them are:

- The example is not interactive; students cannot ask for an example using their own data set, especially if they are confused on corner cases not present in the example

- Teachers will have difficulties preparing the examples that suit the students; complex examples risk confusing the students and contain bugs, while simple examples risk being too obvious for the students

Hand-drawn diagrams solve those problems, since teachers can directly ask students the data set they want to use. However, this method comes with its own set of problems:

- Teachers need to prepare beforehand; inability to explain the material using students' data set, which is arbitrary, can both confuse students and give the perception that the teacher is not knowledgeable. Being knowledgeable is important factor in students' perception of the teacher (Delaney et al, 2010)
- Students' data set have the potential to become too complex and unmanageable during the short time allocated for the section or too trivial/ill-defined. Either way, this might be detrimental to the teaching activity
- Teachers can try to create their own data set instead of using students' data set, but this defeat the purpose of using hand-drawn diagrams; pre-scripted examples would do better in this case.

Lastly, teachers can simply give students links to pre-existing visualization on Data Structures and Algorithms. However, there are problems as well:

- Visualizations are usually not unified, but only done for only a specific algorithms; this means that visualizations does not have the same look and feel, which causes an extra learning curve for students to use the visualizations in the first place
- Different visualizations might use different pseudocode for the same operation, potentially causing confusion among students
- Visualizations might haven't been implemented yet, forcing teachers to resort to the first two methods

This project can improve teaching experience in multiple ways, some of them are:

- Visualization tool

    - Visualization tool allows students to use their own data set without overwhelming the teacher, since the explanation process is automated
    - Visualization tool is guaranteed to implement most materials presented in NUS class, and it will be more geared towards NUS curriculum
    - Visualization tool have the same look and feel, so students will only have to learn using one unified system
- Online judge tool
    - The tool opens the potential of teachers asking impromptu questions during lectures, increasing interactivity
    - Feedback from impromptu questions are available almost instantly, allowing both teachers and students to analyse common pitfalls

In summary, this project can enhance teaching experience by solving problems present in traditional methods and introducing a more interactive method to deliver materials.

Improving Process and Quality of Graded Assessment

According to the revised Bloom's Taxonomy by Krathwohl (Krathwohl, 2002), cognitive learning process can be divided into 6 parts (from lowest level to highest level):

1. Remember: Retrieving relevant knowledge from long-term memory
2. Understand: Determining the meaning of instructional messages, including oral, written, and graphic communication
3. Apply: Carrying out or using a procedure in a given situation
4. Analyze: Breaking material into its constituent parts and detecting how the parts relate to one another and to an overall structure or purpose
5. Evaluate - Making judgments based on criteria and standards
6. Create - Putting elements together to form a novel, coherent whole or make an original product.

Dr. Steven Halim found out that in graded assessments, students will have to be asked questions on the 3 lowest level of the taxonomy: remember, understand, and apply. Answers to these questions are trivial, and a significant majority of the student will be

able to answer the questions correctly, but still need to be asked to identify students who are struggling with the concepts / unqualified to pass the module. However, the process of creating and verifying the answers of such questions is very tedious and repetitive. The presence of an online judge tool would solve this problem by automating the process of creating and verifying the answers of these types of questions. Furthermore, the online judge tool will allow more time during graded assessments to be devoted to discuss questions of higher taxonomy level, such as analyse and evaluate level.

1.4 Individual Objectives

Designing an Extensible System for Long-Term Development

Among the drawbacks of the previous visualization tool, the most serious one is the lack of good software engineering practices, resulting in spaghetti code and unclear separation of concern. Dr. Steven Halim indicates that there has never been any effort to enforce such practices, and past developers are just trying to "follow the standard" set by the original developers. In the long run, this will cause future development to incur high technical debt of trying to fix hidden bugs and compatibility issues.

Due to this fact, one of my two main objectives is to design an extensible system that can be maintained in long-term development. I am responsible for designing the system as a whole and making sure that the main logic of the system is well-structured, mainly by utilizing my knowledge in software engineering from CS2103 and CS3213. It is also my responsibility to document the system, as well as implement the main logic of the system I designed.

Creating a Working Version of Online Judge Tool

Ultimately, the milestone of this project in this academic year is to implement an online judge tool which is usable in NUS classes. I am responsible for implementing the logic features of the online judge tool, particularly on its ability to generate questions and verify answers. I am also responsible for other utilities to enhance the usability of the system, such as the ability to create parameterized test questions and storing students' answers in the database.

2 Literature Review

2.1 Existing Visualization Platforms

Several visualization platforms for Data Structures and Algorithms exist before this project started even back in 2011. Some of the most notable ones are http://nova.umuc.edu/~jarc/idsv/ , http://research.cs.vt.edu/AVresearch/ and http://www.cs.usfca.edu/~galles/visualization/Algorithms.html , with the latter providing visualization on most data structures and algorithms covered in CS2010 and CS1020 (Galles, 2011). More visualization tools can be found on the website http://algoviz.org/ which provides links to various visualization tools (AlgoViz, 2004). Dr. Halim's paper contains an extensive review on these platforms, and will not be discussed further here.

There are several reasons why the visualization tool is developed even though there are already existing systems implemented. One reason is because our visualization tool aims to go beyond the common data structures and algorithms taught until CS2010, up to the exotic data structures and algorithms taught in CS3233. Another reason, and the most important one, is that the visualization tool is just a component of the greater part of the system, which also contains other tools such as the online judge tool. The motivation of this project is to provide a unified look and feel, and thus it is necessary to develop our own visualization tool.

2.2 Existing Online Judge Platforms

Understanding how online learning and testing works currently is also essential for this project, since it helps to gather ideas on how the online judge tool should be created.

At the beginning of the project, I tried to experience how MOOC works by taking an online course in Coursera. My conclusion is that currently online learning is currently not conducted efficiently, mainly due to the low motivation of students to complete the course. This is consistent with the data provided by K. Jordan, which mentions that the

completion rate for most MOOC is lower than 13% (Jordan, 2013). However, this is not entirely relevant to the project this academic year since it is out of the project scope, and for me as well since I am not responsible for the user experience.

Online judge tool itself is not a new concept. There are some articles which describes an automated grading system, such as the Automated Essay Scoring system used by edX as described by Dr. Balfour (Balfour, 2013). Some of the critics of the system are that the system is not human enough for essay grading, and thus is not fair. Hence, this project will only support questions which answers are discrete, which is analogous to the project motivation and objectives. Currently there are 2 systems that closely resemble this project:

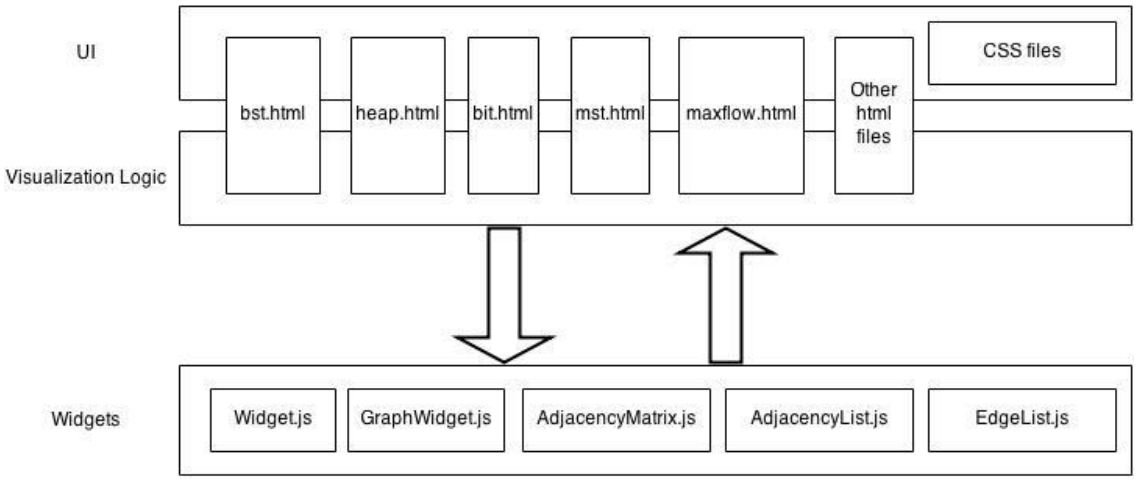
- Wolfram Problem Generator (The Wolfram|Alpha Team, 2013) which generates questions and verify answers for mathematics problems
- TRALKA2 (Nikander et al, 2004) which generates questions and verify answers for computer science problems, and has the same objectives as our system. However, there are more restrictions to use this system compared to our system, particularly because it requires installation and sign-up to use. It is also important to note that the system is last updated in 2009.

Our team have some debates on how to organize online graded assessments using the online judge tool. There are 2 methods which are being considered, which are randomized assessment, in which every students get different set of questions without any human intervention, and parameterized assessment, in which examiners can set the parameters to generate the questions first and every students will get the same set of questions. One of the main benefits of randomized assignment is to prevent cheating, and moreover the system does not suffer cheating problems present with questions pulled from a question bank, as described by B. Scheiner (Scheiner, 2012). We tried to conduct a randomized test using a pool of question bank to the CS2010 students, and to guarantee equal difficulty levels we classified each problems to levels of difficulties, and each student will get the same amount of questions from each level of difficulties. However, a fair amount of students indicated in the CS2010 module survey that they perceive randomization of questions, even with the difficulty level guarantee, is unfair,

particularly because the difficulty level of the assessment will still be different for each student. One such scenario that our team can think of is that a student can be proficient in a certain topic and luckily get questions in that topics alone. Hence, our team decided to implement a parameterized assessment: questions are still randomly generated, but every student will get the same questions although presented in different order. At the very least, questions are guaranteed to be fresh for every assessments.

2.3 Analysis of Previous Team's Visualization Tool

Considering that this project is an ongoing project which has begun 2 years ago, it is important to review the software design before adding a huge feature such as online judge tool. However, due to lack of past documentations, there is no diagrams / description on the software design of the previous team. Below is the block diagram of the system and a sequence diagram describing the previous team's visualization tool from my point of view:



Aside from the lack of structure, the implementation of the old visualization tool itself poses a problem which makes it difficult to implement the online judge tool. Below is a brief description of how the old visualization tool works:

1. Each separate HTML files has a main visualization JavaSript (JS), which will initialize GraphWidget and other backend codes such as Widget, a custom JS

library which supports animation functionalities such as play, pause, stop, and customization of vertex and edge objects such as onclick functions, colors, etc.

2. The main visualization creates three functions for each animation called to be executed by the widgets, which is called *pre*, *algo*, and *end*.
    - pre: Resets the visualization to a pre-execution state. This is used in Replay and Previous functionalities
    - algo: Renders the algorithm visualization and the pseudo-code highlight. This is the main functionalities of the visualization
    - end: Forces the visualization to display the ending state. This also stops the current animation. Used in Stop functionalities
    - autoEnd: This is a fourth parameter which is currently unused
3. The widgets will draw the necessary utilities (play, stop, pause button and the progress bar, FPS display, etc.), hides the other irrelevant buttons, and executes the functions given to it. It first calls *pre*, and then executes *algo*. Users can execute functionalities such as changing FPS, Next, Previous, and Replay, which will not be elaborated in this report.
4. Once the user chooses Stop, the *end* function will be executed and cleanup will follow

From these findings, I identified weaknesses in the old visualization tool which will make it difficult to extend with the online judge tools, particularly due to the following:

- Extra effort required to achieve similar look and feel
  While it might seems that the widgets is the ones doing the rendering, actually it is the functions passed in (*pre, algo, end*) which does the job. This means the main visualization is the one doing the rendering; the widgets are only concerned with providing the necessary infrastructures such as progress bar, and executes the functions passed in. *New tools cannot use existing code to achieve similar look and feel; in order to achieve this, developers have to read the old codebase and implementing the UI part of the tool with similar style.*
- Spaghetti code, low cohesion and tight coupling
  The widgets also contains the implementation of vertex and edge objects which contains algorithm-specific data (e.g. leftChild and rightChild, both only used in

BST-like data structure) instead on focusing on UI data. *Continuing to follow this style will quickly turn the online judge tool codebase unmanageable due to entanglement with visualization tool codebase*.

- Not extensible

  Due to the tight coupling nature of the old visualization tool, adding the online judge tool would be tough, and even more tough to change the underlying library used for the UI to other library. One must at least recode the entire visualization logic to achieve this. *This violates the user requirement which states the project should be extensible.*

- Inherent system limitations

  There are limitations in the system, such as the efficiency of animation. When the user chooses to return to a previous frame, the old library replays the entire animation from the first frame and moves forward to the frame requested by the user. This issue causes even the simplest visualization to be slow to the point that it is noticeable by the user. *Some of these system limitations cannot be fixed without major changes of the old visualization tools.*

In conclusion, major changes are required to the visualization tool before any work can be done to the online judge tool. At minimum, the visualization tool should be able to provide a library which can be used by the online judge tool to achieve similar look and feel.

2.4 Software Engineering

Most of my knowledge on software engineering are obtained from classes such as CS2103 (Software Engineering) and CS3213 (Software System Designs). I used this knowledge to design parts the system using the MVC pattern and several other patterns as seen fit, such as Database Access Pattern. I also employed software design principles described by R. Martin (Martin, 2000) and Microsoft (Microsoft, 2009) in my implementation, especially the following principles:

- Modularization: Different components of the system should be grouped into different classes/object/files/folders

- Separation of concern: Each component or module should not overlap each other in terms of functionality
- Single Responsibility principle: Each component or module should be responsible for only a specific feature or functionality, or aggregation of cohesive functionality.
- Principle of Least Knowledge: A component or object should not know about internal details of other components or objects.
- Don’t repeat yourself (DRY): A component should not re-implement functionalities that are already implemented in other components.
- High Cohesion and Loose Coupling: The system should minimize interdependence and maximize diversity of tasks among classes

3 Software Design Methodology

Below is a description of procedures I follow to design the both the visualization tool and the online judge tool. During development, some of the procedures are not necessarily be conducted sequentially, but revisited as required.

Determine the Requirements

The first step of designing the system as a whole is to determine the requirements; what does the system needs to support and not support, and which functionalities are more important. This provides a clear understanding on the system and the scope of the project as a whole.

Identify the Components

The next step is to divide the system into different components, each of them doing separate tasks. The components are then analysed further to determine their importance in the requirements and their reusability.

Put the Components Together

After identifying the components, I tried to put them together. Here I determined how the components should interact with each other, especially on components that have to be reused in multiple tools.

Identify and Implement an Initial Version of the Critical Component

At this step, it becomes clear which components act as the foundation of the system. Other components will heavily rely on this component to work, and thus the component is implemented with minimum functionalities. In this way, other less critical components can be implemented even by other developers involved in the project.

Create a Proof of Concept of Other Components

Before other developers can assist in other less critical components, I created a proof of concept by implementing one class which uses the critical component, which usually

will enable the system to perform limited features on one data structure or algorithm. Result on this step is visible and is already usable.

Receive User and Developer Feedback

Since the system is usable, it is possible to gather user feedback on functionalities that need to be changed. Developer feedback is also gathered, particularly on whether the developers can implement other classes to perform the same features on different data structure or algorithm.

Implement the Full Version of the Critical Component

Based on the feedback, the critical component is then refined. After the critical component works as desired, all other features can be added safely.

Implement the Rest of the System

The rest of the system can then be implemented with ease, considering the main bulk of the functionalities have been implemented in the critical component.

4 Visualization Tool

4.1 Requirements

Based on the analysis of previous team's work and the long-term vision of the project as a whole, I formulated a list of requirements as follow:

Functional Requirements

1. The visualization tool should include representations of data structures and algorithms taught in the following NUS modules:
    a. CS1020 Data Structures and Algorithms I (Future Works)
    b. CS2010 Data Structures and Algorithms II
    c. CS2020 Data Structures and Algorithms Accelerated (Partially supported)
    d. CS3230 Design and Analysis of Algorithms (Future Works)
    e. CS3233 Competitive Programming (Future Works)
2. All visualizations must be accessible to the user from a central (home) page
3. The visualization tool must be able to show the sequence of steps involved for each operation related to a certain data structure or algorithm
4. The visualization tool must allow the user to control the playback of the sequence of steps for each operation, including the ability to pause, replay, move directly to a particular step in the sequence, and control the playback speed
5. The visualization tool must provide an explanation or description of each step in each operation
6. The visualization tool must provide pseudocode tracing for each operation
7. The visualization tool should allow users to build instances of provided data structures and representations of algorithms using their own datasets, instead of relying on a hard-coded dataset

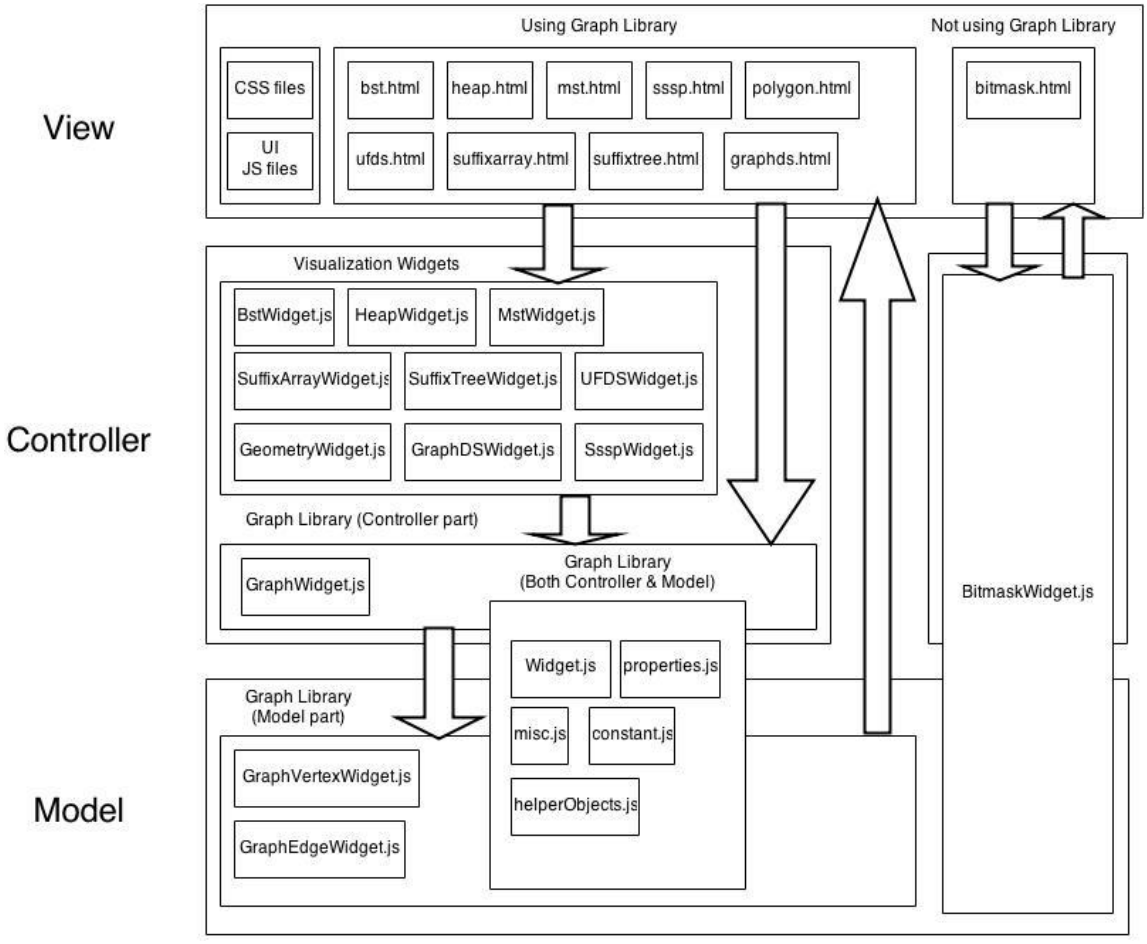
Non-Functional Requirements

1. Performance: All operations and animations should occur with minimal delay
2. Extensibility: The system should be able to accommodate additional secondary features in the future

3. Compatibility: the visualization tool must be able to function fully in the latest version of Google Chrome (Version 33). If possible, it should work on Mozilla Firefox Version 25+ and Safari Version 6+ as well. It should be usable on tablets as well as laptops, on screens with a resolution of at least 1024x768 px.
4. Usability: the visualization tool should be reasonably easy to learn, easy to use, and enjoyable to use

4.2 Software Design

In response to the problems in the old visualization tool, a new visualization tool is created, which is the one currently used in the system. The new tool is made with long-term development in mind, and thus follows the MVC structure. Below is the block diagram of the visualization tool.



There are 3 main part of the new visualization tool:

- Graph Library

  This library defines graph objects which represents the graph described by the visualization tool. The library takes in information on what does the visualization tool should show in the form of “Graph State” objects, and manipulates the DOM objects to display the information.
- Visualization Widgets

  These widgets contains the logic of the data structures and algorithms visualized by the tool. It is equivalent to the “Visualization Logic” in the old visualization tool, only that it does not directly tell the UI on how to draw the graph. It passes “Graph State” objects to the Graph Library which information will be reflected on the UI.
- “Graph State” Object (not shown in the block diagram)

  It is a JS object containing information on what to draw on the UI. This object is passed into the Graph Library to be drawn on the UI.

Structure of “Graph State” Object:

- “vl”: JS object with vertex ID as keys and JS objects containing corresponding attributes as a value
  - Compulsory attributes:
    - cx : X-coordinate of center of vertex
    - cy : Y-coordinate of center of vertex
    - text : Text contained inside the vertex
    - state : Defines the CSS attributes of the DOM objects which are parts of the vertex, as defined in *properties.js*
  - Optional attributes (overrides attributes defined in “state”):
    - inner-r : Customize the vertex's inner radius (Circular)
    - outer-r : Customize the vertex's outer radius (Circular)
    - inner-w : Customize the vertex's inner width (Rectangular)
    - outer-w : Customize the vertex's outer width (Rectangular)
    - inner-h : Customize the vertex's inner height (Rectangular)
    - outer-h : Customize the vertex's outer height (Rectangular)

        - inner-stroke-width : Customize the vertex's inner stroke width
        - outer-stroke-width : Customize the vertex's outer stroke width
        - text-font-size : Customize the vertex text's font size
- "el": JS object with edge ID as keys and JS objects containing corresponding attributes as a value
    - Compulsory attributes:
        - vertexA: id of vertex A
        - vertexB: id of vertex B
        - type: undirected, directed, or bidirected (direction: A->B)
        - weight: The edge's weight
        - state: Defines the CSS attributes of the DOM objects which are parts of the edge, as defined in *properties.js*
        - animateHighlighted: Determines whether highlighted animation should be played. Boolean
    - Optional attributes:
        - displayWeight: Determines whether weight should be shown. Boolean

How the Visualization Widgets works:

1. The visualization widget receives a user input stating that it needs to visualize a certain function on the data structure / algorithm.
2. The visualization widget creates "Graph State" objects which describes the state of the graph from initial state to final state.
3. All of the information are passed into GraphWidget.js, which will manage how these information will be shown to the user

How the Graph Library works:

1. The graph library receives an array of "Graph State" objects
2. The graph library begins to display graph as described the "Graph State" objects, starting from index 0 to the last index.
3. At any point of time, user can directly manipulate with the displaying mechanism, such as by:

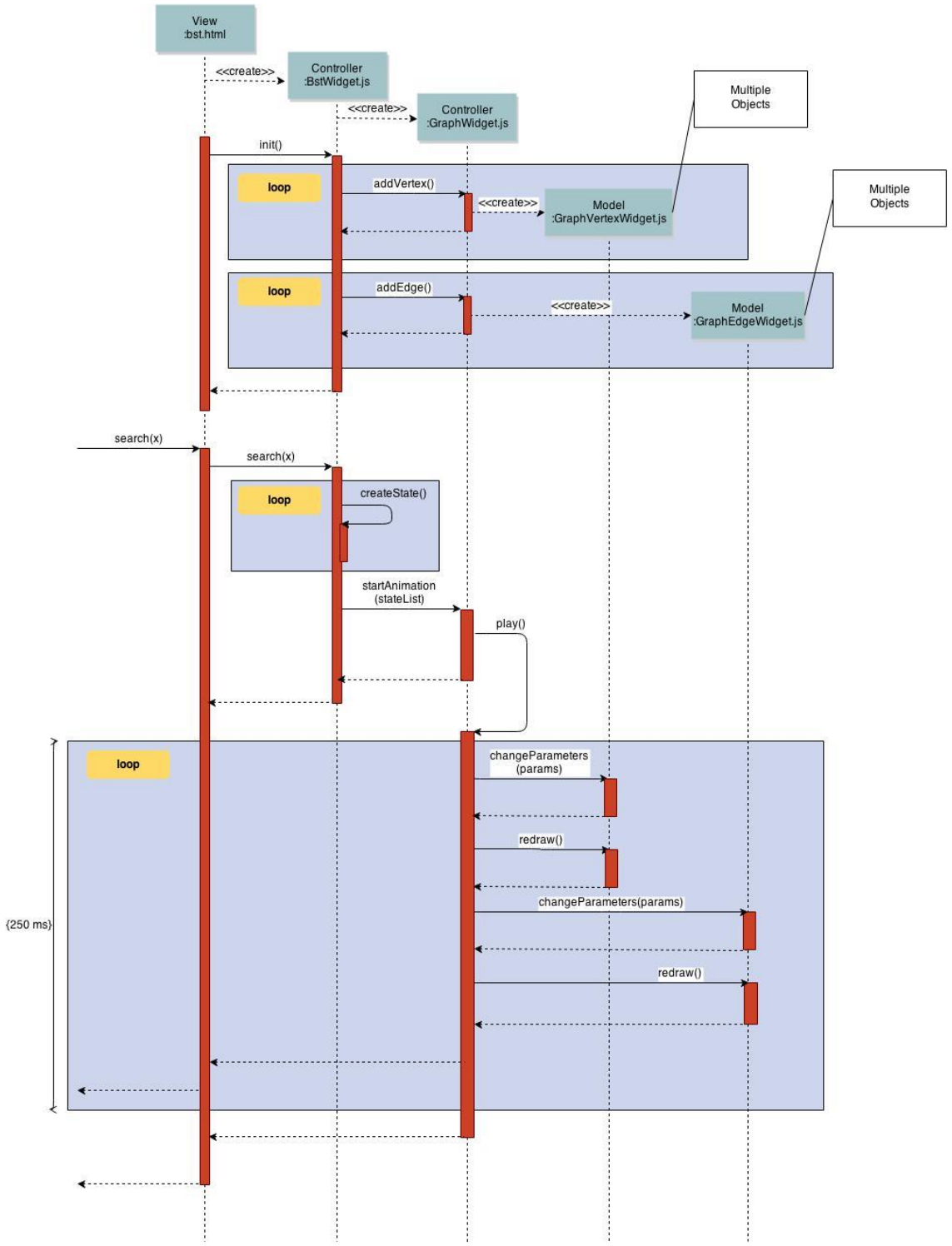
1. Pausing the animation: Stop the display of next “Graph State” object until being resumed
2. Resuming the animation: Resume the display of next “Graph State” object from the last state
3. Jump to iteration x: Jump the animation to the xth “Graph State” object
4. Modify the speed of animation: Make it faster or slower. Default is 4 fps.
5. Stop the animation: Jump to the last state and clears the “Graph State” object array, which effectively prevents users from returning to previous frames again.

One exception to the visualization tool is the Bitmask visualization, which does not use the graph library. This is due to several factors:

- During the development of BitmaskWidget, the Graph Library is still in infancy stage
- BitmaskWidget didn’t use any graph features, because it is not a graph

The problem of restricting visualization to graph structures also poses a problem during the development of SuffixArrayWidget. The team cannot provide a perfect solution at that time, due to the developer of SuffixArrayWidget, Duy, graduating soon. Since the Graph Library is already established, I added temporary features such as rectangular vertex to mask the problem. Better solution to this limitation will be addressed in future works section.

Below is a sequence diagram showing how the parts using Graph Library works to visualize “search(x)” in a BST. Over here, it is assumed that the users do not use any controller functionalities from GraphWidget.


View
:bst.html
<<create>>
Controller
:BstWidget.js
<<create>>
Controller
:GraphWidget.js
Multiple
Objects
init()
loop
addVertex()
<<create>>
Model
:GraphVertexWidget.js
Multiple
Objects
loop
addEdge()
<<create>>
Model
:GraphEdgeWidget.js
search(x)
search(x)
loop
createState()
startAnimation
(stateList)
play()
loop
changeParameters
(params)
redraw()
changeParameters(params)
redraw()
{250 ms}

4.3 Strengths over previous team's visualization tool

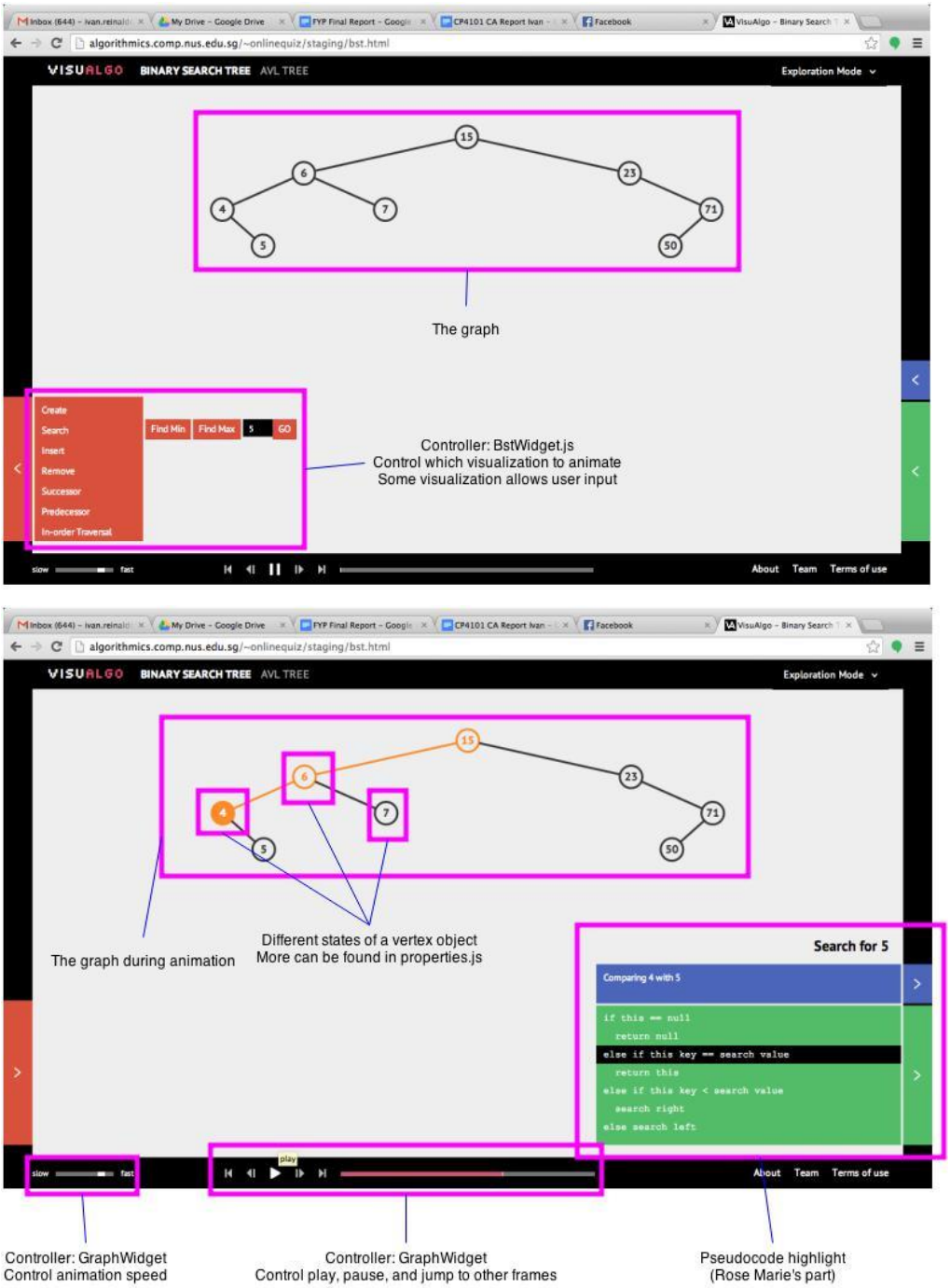
- Modularization and decoupling of the tool
  The new graph library only contains the rendering information, decoupling visualization logic from UI information. Furthermore, the graph library directly manipulates the DOM objects, masking all rendering details from the visualization logic. *The codebase is more manageable, with every parts of the tool doing specific tasks.*
- Reusable graph library to achieve similar look and feel
  Graph library can function independently from visualization widgets; as long as an array of "Graph State" objects is supplied, it can display graphs on the UI. *This means that the online judge tool and other future tools can use the graph library to achieve similar look and feel.*
- Extensible
  Decoupling of the visualization tool makes it much easier to add online judge tool and any other tools; just make sure that the new tools can use the graph library. Furthermore, porting to new underlying UI library is easily done because the only files that need to be modified are GraphVertexWidget, GraphEdgeWidget, and GraphWidget. *This fulfills the user requirement which states the project should be extensible.*
- Fixed system limitations in old visualization tool
  The new visualization tool is developed with the system limitations of the previous visualization tool in mind, and extra caution is made to not reintroduce them again. For example, animations are now more efficient because the new visualization tool can jump between frames in O(1) time complexity; all that needs to be done is to load the "Graph State" object for that frame. *System limitations present on the old visualization tools are no longer present in the new visualization tool.*

Aside from the benefits, there are also new issues introduced by the new visualization tool. These will be addressed in future works section.

4.4 Proof of Concept

In order to demonstrate how the graph library works, I created two visualization widgets using the library. They are visualizations of BST and MST.

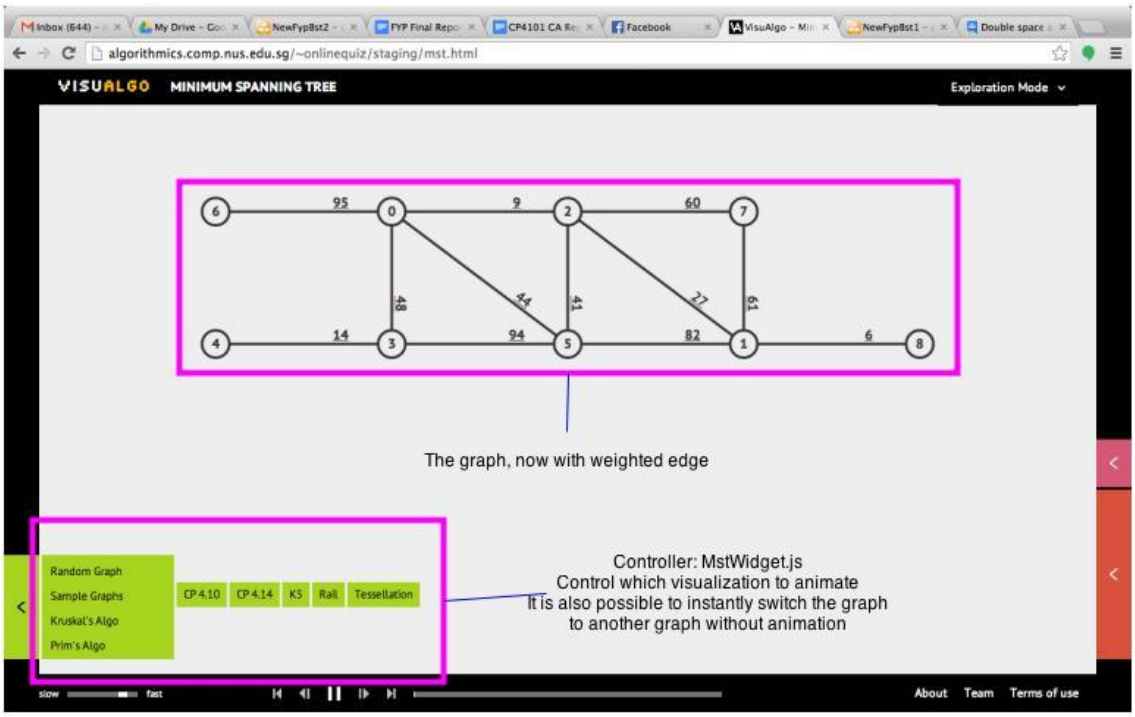
BST Visualization

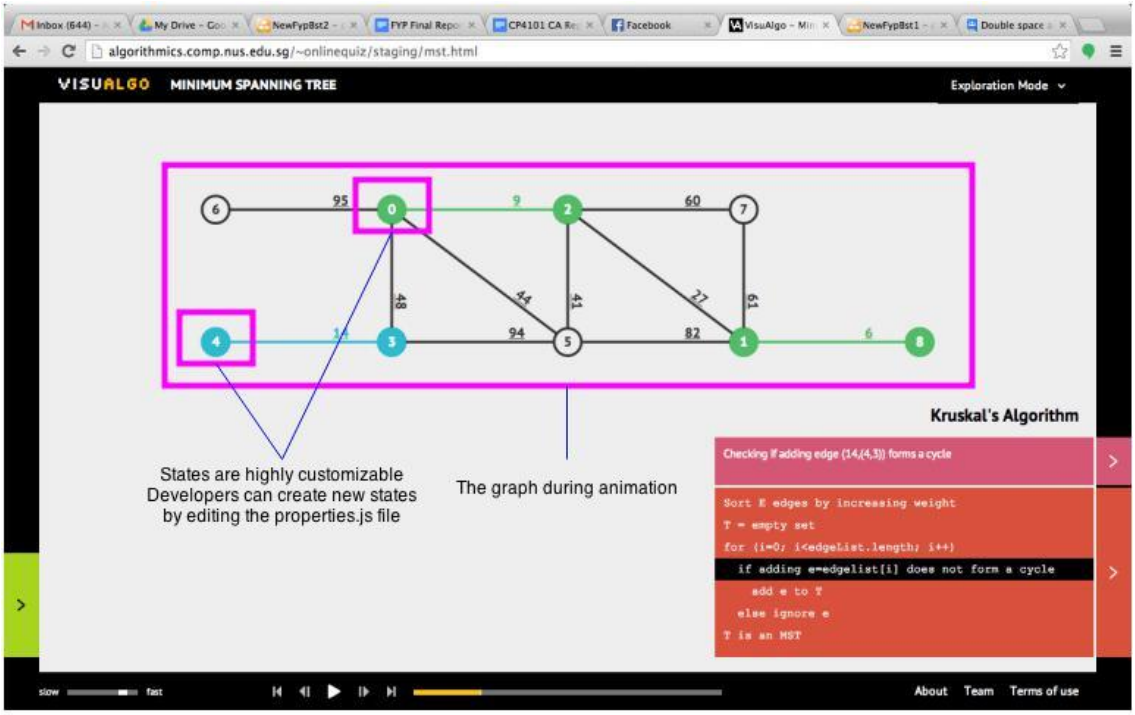


List of functionalities implemented:

- AVL Tree, which is a balanced BST alternative
- Creating a new BST/AVL based on either user input or random number generator; rules can also be set in the random number generator, such as generating BST that is skewed to left/right side
- Insertion of new vertex/vertices and its animation
- Deletion of existing vertex/vertices and its animation
- Successor/predecessor animation of a vertex inside the tree
- In-order traversal animation of the tree

MST Visualization





List of functionalities implemented:

- Graph generator from predefined templates (weight and direction of edges are randomized)
- Prim algorithm animation
- Kruskal algorithm animation

4.5 User Feedback

Below are some selected comments on the visualization tool from the official CS2010 AY2013/2014 Semester 1 module survey:

- Positive
    - I really like the new visualization tool as it really helps solve some of my problems in imagining how an algorithm works :)
    - Clean and simple UI. The visualization tool is clearer than the one printed on the past year papers.
    - Visualization tool is a great step forward for learning.

    - The new teaching aid (WeNeedAName) gives a lovely visual walkthrough on how the algorithms covered work. Excellent utility. The module also uses the above teaching tool to help facilitate electronic examinations, a welcomed improvement that should have been done a long time ago. Quite frankly, as a computing society (let alone a SCHOOL of computing), we should be moving away from pen-and-paper based examinations. Other modules, such as CS1010 and CS1020, should utilize this tool to help teach their own respective data structures and algorithms.
    - Visualization tool is extremely well made and useful for self-study.
    - The lectures can be a bit slow some times, but overall, it is very easy to understand what is being taught, partly due to the visualization tool.
- Negative
    - Should never get rid of the step-by-step lecture slides even if the visualization tool is completed. I didn't use the visualization tool because I felt the tool was distracting and the lecture slides were clearer and straight to the point.

Overall, there is only a total of 1 negative comment on both mid-term survey and end of module survey. All other comments on the visualization tool are positive comments. “WeNeedAName” is the temporary name of this project before VisuAlgo.

5 Online Judge Tool

5.1 Requirements

Based on the long-term vision of the project as a whole, I formulated a list of requirements as follow:

Functional Requirement

1. The online judge tool should be able to generate questions on data structures and algorithms taught in the following NUS modules:
    a. CS1020 Data Structures and Algorithms I (Future Works)
    b. CS2010 Data Structures and Algorithms II
    c. CS2010 Data Structures and Algorithms Accelerated (Partially supported)
    d. CS3230 Design and Analysis of Algorithms (Future Works)
    e. CS3233 Competitive Programming (Future Works)
2. The online judge tool must be able to display the questions it generates to the user clearly, both in terms of question phrasing and visual display of data structures
3. The online judge tool must be able to handle a predefined set of types of user input, and ensure that user input is recorded and sent to the server with 100% accuracy. The types of user input it should handle include but are not limited to:
    a. Vertex selection (single and multiple, ordered and unordered)
    b. Edge selection (single and multiple, ordered and unordered)
    c. Number input
    d. Multiple choice options
    e. "No answer" option where applicable
4. The online judge tool must always show the user his/her current answer to each question where applicable
5. The online judge tool should allow the user to navigate between the test questions
6. For graded online tests, the online judge must constantly display to the user the time remaining for the test
7. The online judge tool must be able to verify the correctness of answers to the questions it generates with 100% accuracy

8. The online judge tool must be able to display the answers to the questions it generates to the users
9. The online judge tool must be able to grade the responses for the question sets it generates and display the resulting score to the user
10. The online judge tool should allow the test administrator to generate a parameterised question set to be used in a graded online test
11. The online judge tool should allow the test administrator to control student access to the graded online test and answer key, as well as monitor and record the results of the test

Non-Functional Requirement

1. Stability: the online judge tool must always be able to run without bugs, especially during graded tests
2. Compatibility: the online judge tool must be able to function fully in the latest version of Google Chrome (Version 33). It should be usable on screens with a resolution of at least 1024x768 px.
3. Usability: the online judge tool should be instantly usable (practically no learning curve), and easy to use
4. Security: students should not be able to gain access to the test questions before the test, the test answers before or during the test, allow other students to take the test for them, give themselves more time, or change recorded test scores
5. Extensibility: the system should be able to accommodate additional secondary features in the future

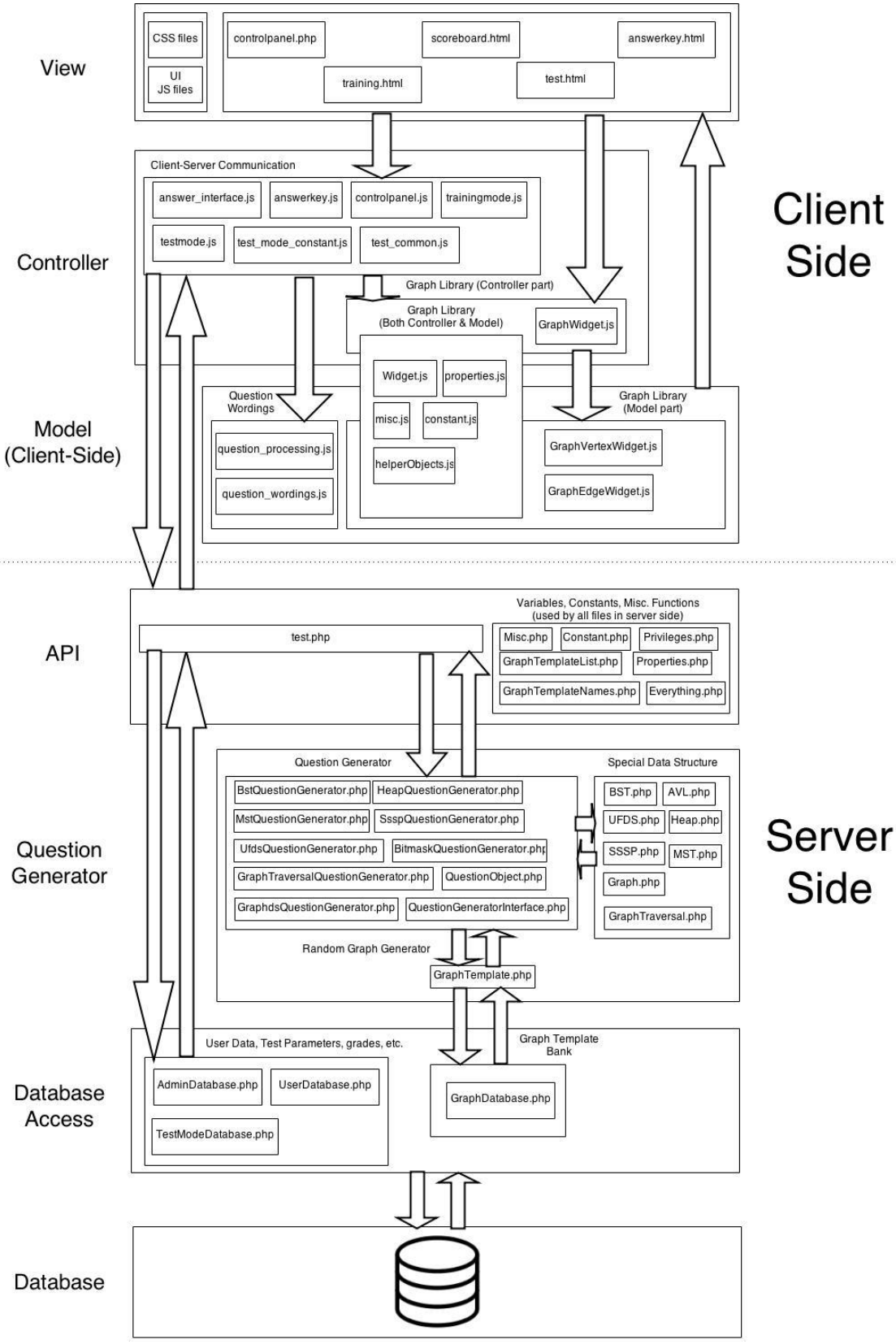
5.2 Software Design

The online judge tool itself has two functionalities, which are:

- Training mode: Generate random questions for users to practice on
  Defining features:
  - Questions generated are randomly generated (mostly by utilizing PHP random number generator), and the system made no effort to recall the questions it have generated in the past

    - User can choose the topics of the questions; questions being generated will only ask user questions on the selected topics
    - Grading result will be released almost instantly, and user can immediately see the answer key of the generated questions
    - Currently, user cannot modify the practice settings, such as the duration of practice and amount of questions
- Test mode: Generate parameterized questions for grading purposes
  Defining features:
    - Questions generated are parameterized; the examiner can set the parameters of the questions, such as the seed, topics, and question amount, and the system will generate the questions according to the parameters. More will be added in future works.
    - Examiner can modify test settings, such as the duration of test and amount of questions
    - Grading result is released for both the user and the examiner almost instantly, and examiner can allow/prevent everyone else to access the answer key
    - The test settings and parameters required to generate the question set are stored inside the database

Similar to the new visualization tool, this tool is implemented with long-term development, and thus uses software design principles and good coding practices while still fulfilling the requirements. Below is the block diagram of the online judge tool.


View
CSS files
UI JS files
controlpanel.php
scoreboard.html
answerkey.html
training.html
test.html
Client-Server Communication
answer_interface.js
answerkey.js
controlpanel.js
trainingmode.js
testmode.js
test_mode_constant.js
test_common.js
Controller
Graph Library (Controller part)
Graph Library (Both Controller & Model)
GraphWidget.js
Widget.js
properties.js
misc.js
constant.js
helperObjects.js
Question Wordings
question_processing.js
question_wordings.js
Graph Library (Model part)
GraphVertexWidget.js
GraphEdgeWidget.js
Model (Client-Side)
Client Side
API
test.php
Variables, Constants, Misc. Functions (used by all files in server side)
Misc.php
Constant.php
Privileges.php
GraphTemplateList.php
Properties.php
GraphTemplateNames.php
Everything.php
Question Generator
BstQuestionGenerator.php
HeapQuestionGenerator.php
MstQuestionGenerator.php
SsspQuestionGenerator.php
UfdsQuestionGenerator.php
BitmaskQuestionGenerator.php
GraphTraversalQuestionGenerator.php
QuestionObject.php
GraphdsQuestionGenerator.php
QuestionGeneratorInterface.php
Special Data Structure
BST.php
AVL.php
UFDS.php
Heap.php
SSSP.php
MST.php
Graph.php
GraphTraversal.php
Random Graph Generator
GraphTemplate.php
Question Generator
Server Side
User Data, Test Parameters, grades, etc.
AdminDatabase.php
UserDatabase.php
TestModeDatabase.php
Graph Template Bank
GraphDatabase.php
Database Access
Database

There are 4 main parts of the online judge tool:

- Question generator and verifier

  This collection of classes generate questions of corresponding data structure and algorithms. It also has the functionalities to verify the answers to questions it generated. Questions generated this way are pseudo-random, controlled only by parameters. It is connected to the special data structure classes, which is the same data structure as what it represents except with their output altered (e.g. if we search a variable inside the BST in BST.php, it will return the sequence of vertices checked instead of the standard boolean true/false).

  If the question generator requires the use of a graph, it can ask the random graph generator to generate a random graph for it.
- API

  The API is called by the client-side controller using GET requests, mainly to get generated questions and to send student answers. Inputs sent from the client side are not immediately stored into the database; they are either processed first by the database access classes or the question generator classes if the input is answers to question sets.
- Database access

  Classes inside database access provides a simple layer of separation between the database and the rest of the server-side classes. Only the database access classes directly connect with the database, and thus the schema is hidden from the rest of the classes.
- Database

  The database stores mainly two types of data, which are user related data and question generator related data. More on this and the schema itself will be explained in the next few sections

How the question generator works

- Generating questions:
  1. The question generator object gets a request to generate a certain amount of questions; different topics of questions are obtained by sending the request to corresponding classes of the question generator

2. The question generator generates questions from the pool of question types implemented in it until it reaches the desired amount
3. The question generator returns an array of QuestionObject class, which will be converted to JavaScript Object Notation (JSON) format by the API

- Verifying answers:
    1. The question generator will need to re-generate the questions first as described above
    2. The question generator gets a request to verify the answer and the question object related to the answer
    3. The question generator will verify whether the answer provided is indeed the answer to the question object passed in, and will return true/false
- Obtaining answers to a certain question set:
    1. The question generator will need to re-generate the questions first as described above
    2. The question generator gets a request to get the answer of the question object provided
    3. The question generator will return the answer to that question object

API

All of the API calls are HTTP GET requests.

Parameter to differentiate requests: mode.

Modes are implemented as constants on both the client side and the server side.

- MODE_GENERATE_QUESTIONS: Generate questions
    - Parameters:
        - qAmt: Question amount (integer)
        - seed: Seed to generate questions (integer)
        - topics: List of allowed topics (list of strings, implemented as constants)
    - Output: Array of JS objects containing the questions

      Structure of JS object:

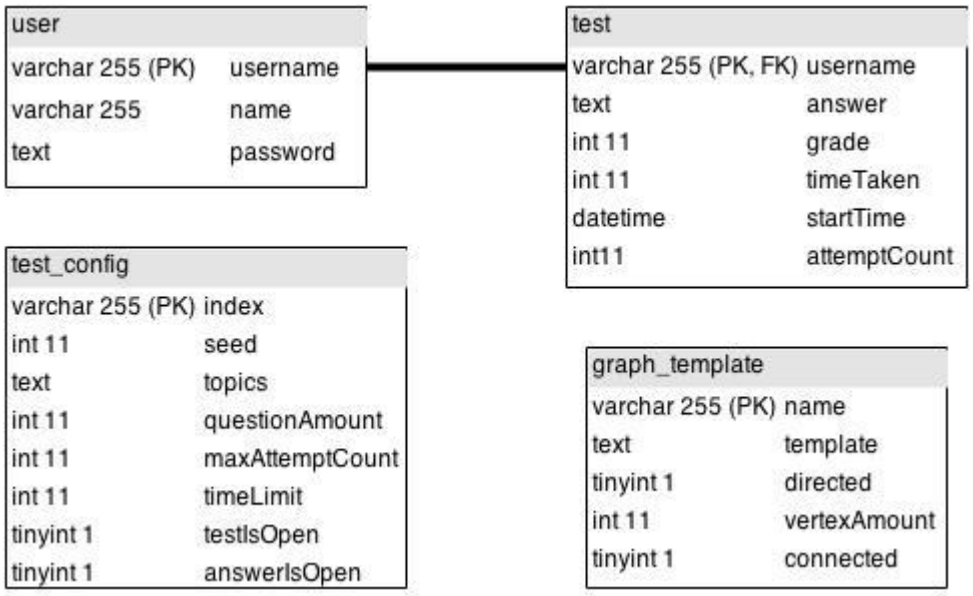
        - “qTopic": Question topic (string, implemented as constants)
        - "qType": Question type (string, implemented as constants)
        - "qParams": Question parameters, such as the values asked in the question and the subtype of the question (JS object, structure varies)
            - “subtype”: Question subtypes. Always present inside qParams (string, implemented as constants)
        - "aType": Answer type, such as whether the answer is MCQ, fill in the blanks, or click one/several vertices/edges (string, implemented as constants)
        - "aParams" => Contains MCQ choices in case of MCQ-type questions, otherwise is null (JS object)
        - "aAmt": Answer amount; some questions requires users to click multiple amount of vertices / edges which fulfills a certain criteria (int)
        - "ordered": If there are multiple answers, this indicates whether the order of which the answers are selected matters (boolean)
        - "allowNoAnswer": Whether the question should show the “No Answer” option (boolean)
        - "graphState": The “Graph State” object to be displayed in the UI
- MODE_CHECK_ANSWERS: Check the answers to a set of questions
    - Parameters:
        - ans: Answers to the question
        - qAmt: Question amount (integer)
        - seed: Seed to generate questions (integer)
        - topics: List of allowed topics (list of strings, implemented as constants)
    - Output: The amount of correct answer
- MODE_GET_ANSWERS: Get the answers to a set of questions
    - Parameters:
        - ans: Answers to the question
        - qAmt: Question amount (integer)

        - seed: Seed to generate questions (integer)
        - topics: List of allowed topics (list of strings, implemented as constants)
    - Output: Array containing the answers
    - Postcondition: If the student answers correctly for a certain question, a string constant “CORRECT” will be sent instead of the correct answer for that question
- MODE_LOGIN: Login to the system. Currently only used in test mode
    - Parameters:
        - username: User’s username
        - password: User’s password
    - Output: 1 (successful) or 0 (unsuccessful)
- MODE_CHECK_TEST_OPEN: Checks whether the user can access the test mode questions and answer key
    - Parameters: None
    - Output: Array of 1 or 0 containing the information [test is open, answer is open]
- MODE_TEST_GENERATE_QUESTIONS: Retrieves the questions for the test mode
    - Parameters:
        - username: User’s username
        - password: User’s password
        - type: Determines whether the question generated is being used to display the test or the answer key
    - Output: Same format as MODE_GENERATE_QUESTIONS
    - Precondition: Test questions is accessible by the user
    - Postcondition: If the question generated is being used to display the test questions, the user’s attempt count will be updated
- MODE_TEST_SUBMIT: Submits the test mode answers and get the user’s grade
    - Parameters:
        - username: User’s username

        - password: User's password
        - ans: The user's answers
    - Output: Amount of correct answers
    - Precondition: User is currently doing the test
    - Postcondition: The answers and other associated data will be recorded in the database
- MODE_TEST_GET_INFO: Get the test utility data, such as time remaining
    - Parameters:
        - username: User's username
        - password: User's password
    - Output: [time elapsed (seconds), user's real name, time limit (seconds)]
    - Precondition: User is currently doing the test
- MODE_TEST_GET_ANSWERS: Get the test's answer key
    - Parameters:
        - username: User's username
        - password: User's password
    - Output: Array containing the answers
    - Precondition: Answer key is accessible by the user
- MODE_TEST_GET_STUDENT_ANSWERS: Get the user's answer to the test
    - Parameters:
        - username: User's username
        - password: User's password
    - Output: Array containing the answers
    - Precondition: User have done the test
- MODE_ADMIN: Login as admin
    - Parameters:
        - password: Admin password
    - Output: 1 (successful) or 0 (unsuccessful)
- MODE_ADMIN_GET_CONFIG: Get the current test parameters
    - Parameters:
        - password: Admin password

    - Output: JS object of the test configurations (see Database Schema section, test_config table)
- MODE_ADMIN_EDIT_CONFIG: Edit the test parameters
    - Parameters:
        - password: Admin password
    - Output: None
    - Postcondition: Test configurations is updated in test_config table
- MODE_ADMIN_RESET_ATTEMPT: Reset the attempt counter of a user
    - Parameters:
        - password: Admin password
        - username: User's username
    - Output: 1 (successful) or 0 (unsuccessful)
- MODE_GET_SCOREBOARD: Gets the scoreboard of the test
    - Parameters: None
    - Output: Array of JS object containing the scoreboard information
        - username: The user's username
        - name: The user's real name
        - grade: The user's grade
        - questionAmount: Total number of questions aksed in the test
        - timeTaken: Time taken to complete the test

Database Schema

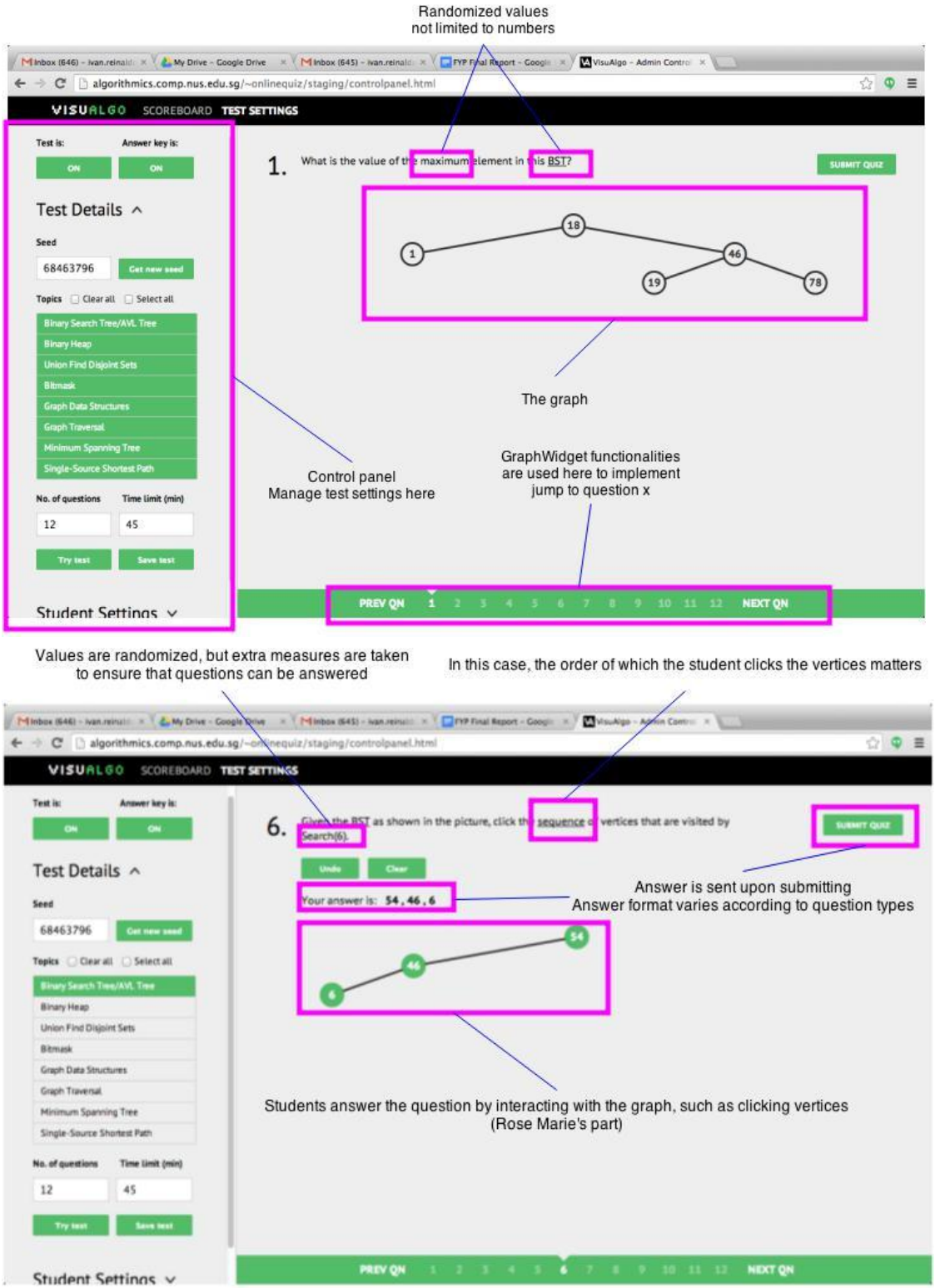


- user: stores information related to user accounts; currently there's no admin account, and admin password is hardcoded in the system
    - username: the username of the user (Primary Key)
    - name: the real name of the user
    - password: the password associated with the username; currently not encrypted
- test: stores information related to test mode results of the users
    - username: the username of the user (Primary Key, Foreign Key to 'user' table)
    - answer: the user's answer for the test in the form of serialized PHP array
    - grade: the user's grade for the test
    - timeTaken: how long does it takes for the user to complete the test in seconds; calculations are done using server time
    - startTime: when does the user starts taking the test; calculations are done using server time
    - attemptCount: how many times have the user attempted the test; can be reset by the examiner
- test_config: stores parameters to generate the test mode questions and settings

    - index: the index of the config; doesn't mean much because there is only one table entry (Primary Key)
    - seed: the seed used to generate the test questions
    - topics: the topics of the test questions in the form of serialized PHP array
    - questionAmount: total amount of the test questions
    - maxAttemptCount: maximum amount of attempt that a user can take for the test
    - timeLimit: time limit of the test in seconds
    - testIsOpen: determines whether users can access the test
    - answerIsOpen: determines whether users can access the answer key
- graph_template: stores graph templates used in random graph generator
    - name: the name of the graph template (Primary Key)
    - template: the graph template in the form of "Graph State" object, stored as serialized PHP array
    - directed: indicates whether the graph in the template is a directed graph
    - vertexAmount: how many vertices does the graph described by the graph template has
    - connected: indicates whether the graph in the template is a connected graph

5.3 Proof of Concept

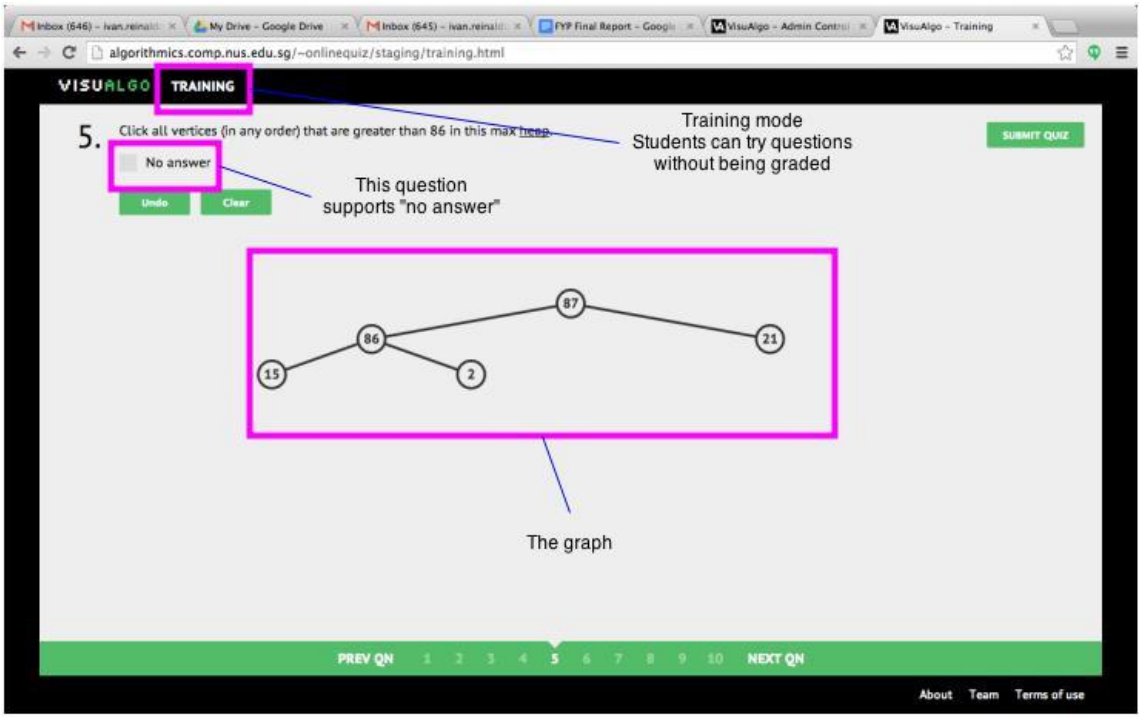
Binary Search Tree Question Generator

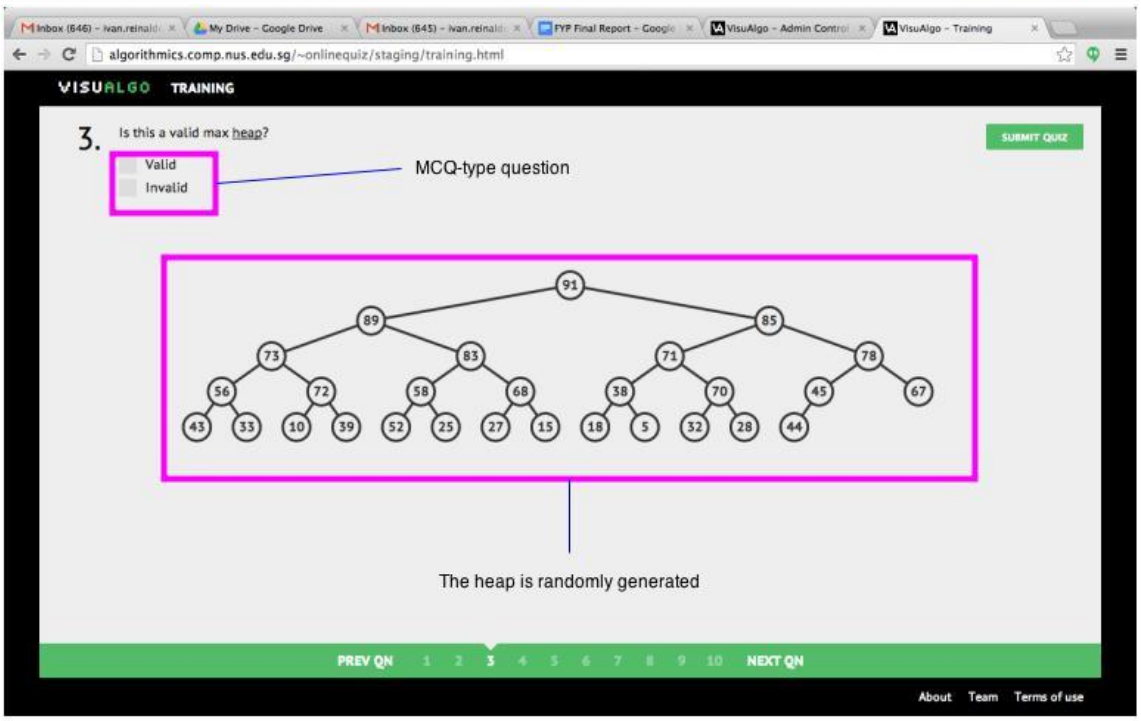


List of question types implemented:

- Click the sequence of vertices from the root when the search for random integer X is executed on a BST containing N [1..10] integers
- Click one vertex with the minimum or maximum value of a BST containing N [1..10] integers (More likely to generate “linked-list” BST)
- Click the sequence of vertices that will be executed by the “Successor(X) / Predecessor(X)” in the BST containing N [1..10] integers
- Click the sequence of vertices in the BST according to the inorder/preorder/postorder traversal starting from the root
- Given a BST containing N [1..10] integers, decide if it is balanced according to AVL tree criteria
- Given a binary tree structure containing N [2..20] integers, determine whether it is a BST
- Given a BST containing N [1..10] integers, determine the height of this BST
- Given a BST containing N [1..10] integers, determine the K [1..N] th smallest element
- Given a BST containing N [1..10] integers, click all the leaf/root/internal vertices. Leaf is defined as vertex with no children. Root is defined as vertex with no parent. Internal vertices are the rest.
- Given an AVL tree containing N [1..10] integers, insert/delete M[1...3] vertices such that Y (0/1/2) Left/Right/LeftRight/RightLeft rotation occurs (Currently disabled due to the question generator’s limitations in generating correct answer)

Binary Heap Question Generator





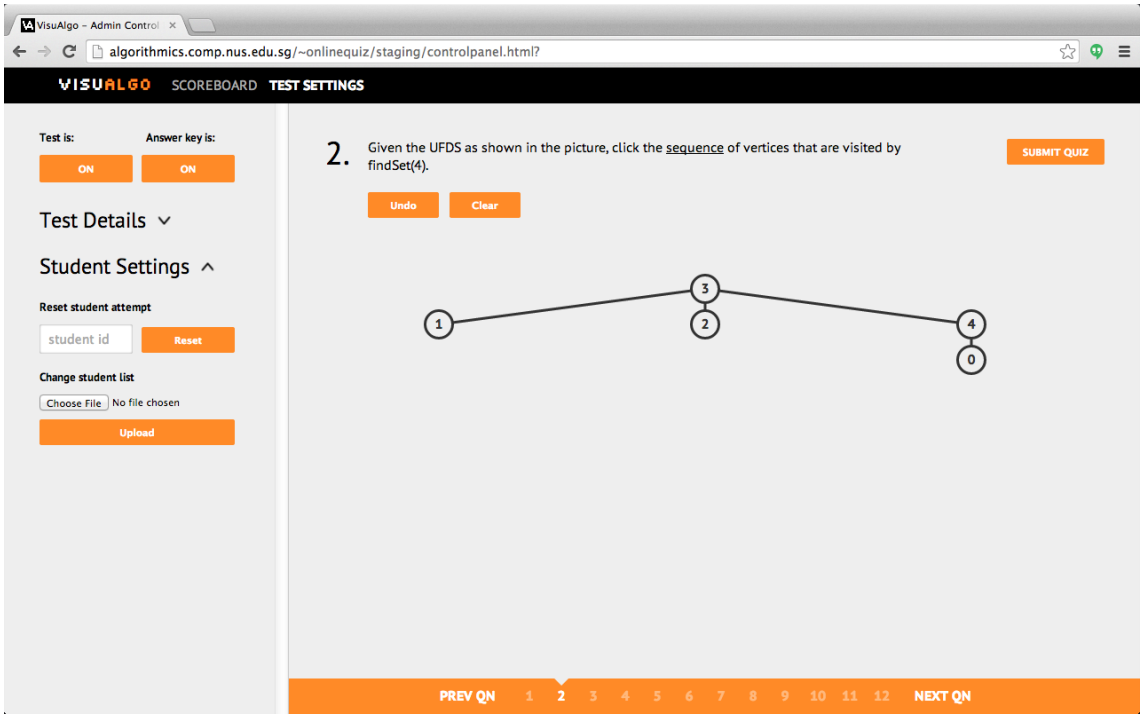
List of question types implemented:

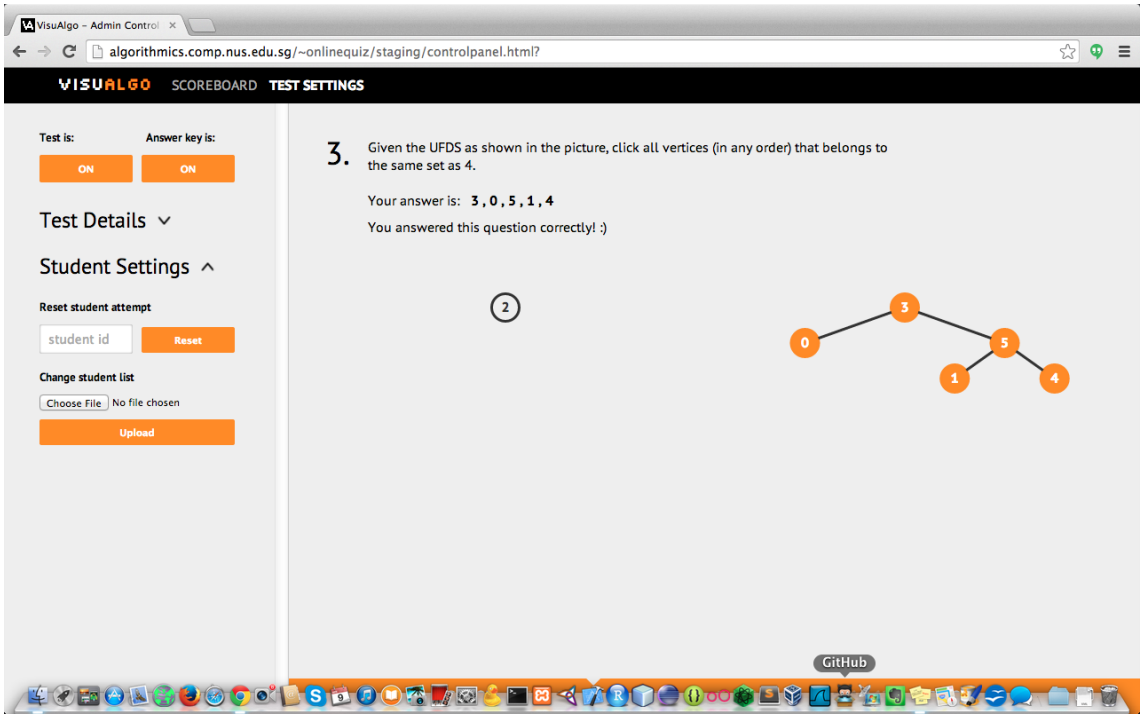
- Given a min/max binary heap containing N [1..31] integers, insert a new integer X [1..99]. This new integer has been inserted at the appropriate new leaf. Click a

sequence of shift up operations (vertices that will be swapped with the inserted vertex) or click a special ‘No swap’ button if there is no swap at all

- Given a min/max binary heap containing N [1..31] integers , we want to extract the min/max item. The bottom rightmost existing leaf has been moved to the root and the heap size has been decreased by one. Click a sequence of shift down operations or click a special ‘No swap’ button if there is no swap at all
- Given a min/max binary heap containing N [16..31] integers , click the leaf/root/internal vertices. Leaf is defined as vertex with no children. Root is defined as vertex with no parent. Internal vertices are the rest
- Given a min/max binary heap containing N [16..31] integers, click the left child/right child/parent of a certain existing vertex
- Given a min/max binary heap containing N [16..31] integers , click all vertices that are less than/greater than X
- Given a min/max binary heap containing N [20..31] integers , we want to perform partial heap sort. After K [4/5*N..N-1]-th step, what will remain in the min/max binary heap?
- Given a starting array that may or may not be a min/max binary heap containing N [1..31] integers, we want to perform the O(n) build heap. Select the root vertices of any sub tree that still violate the min/max heap property and shift down will be performed on those vertices
- Given a min/max binary heap containing N [1..31] integers, determine whether the heap is a min heap or a max heap

Union Find



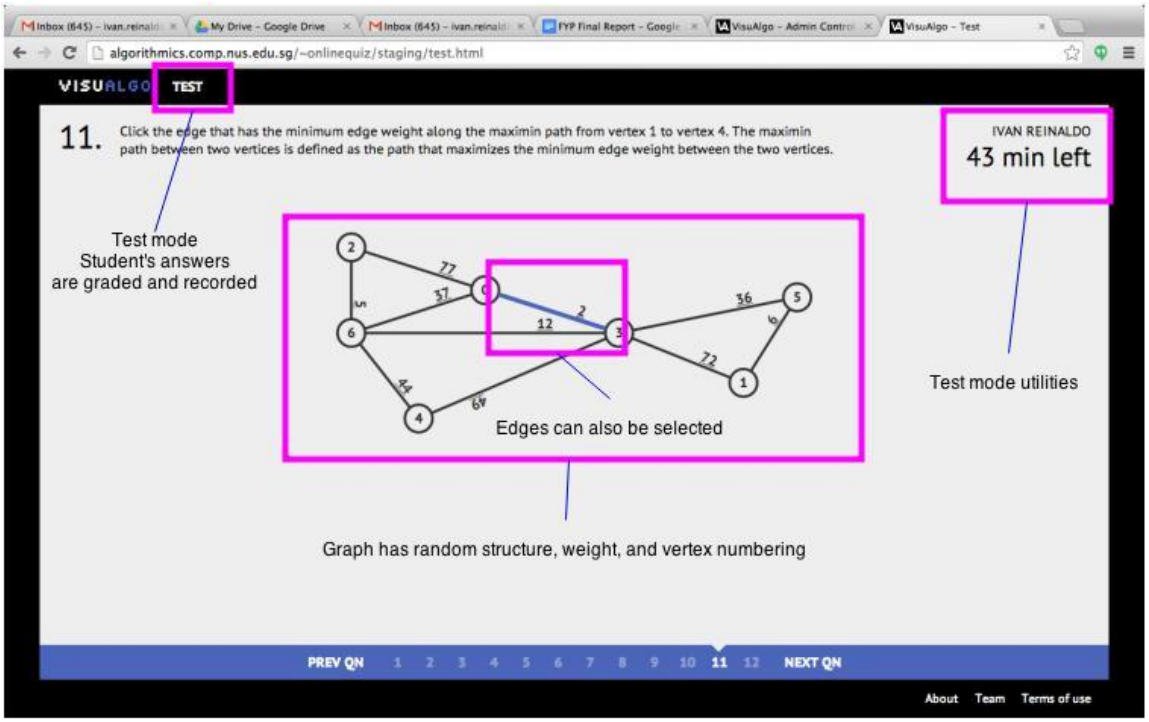


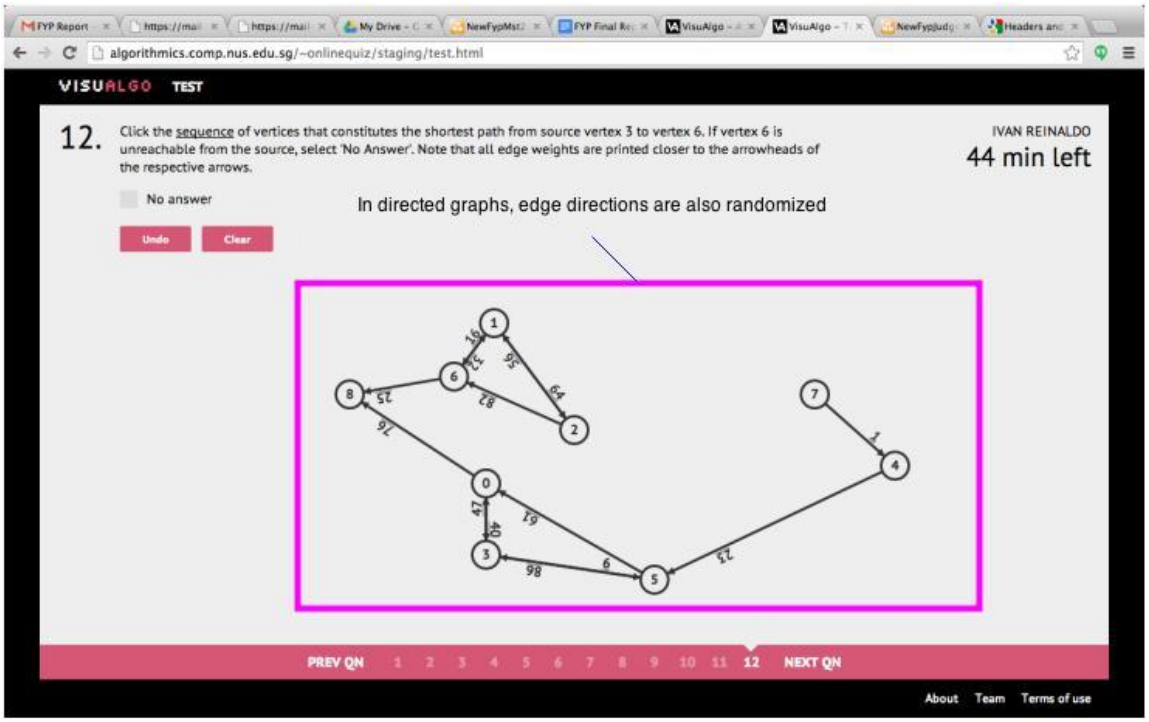
List of question types implemented:

- Given a UFDS tree of N[1..15] integers, click all vertices that belongs to the same set as vertex X

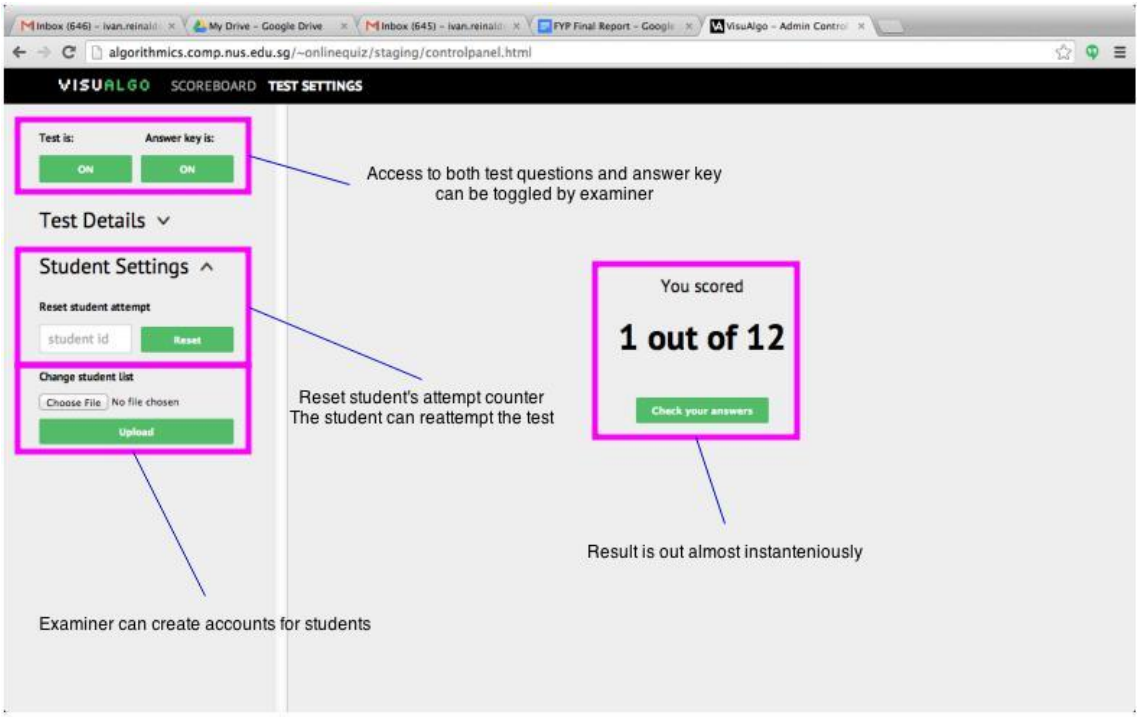
- Given a UFDS tree of N[1..15] integers, click the sequence of vertices that are visited by findSet(X)
- Given a UFDS tree of N[1..15] integers, click all vertices X which causes the UFDS structure to change when findSet(X) is called

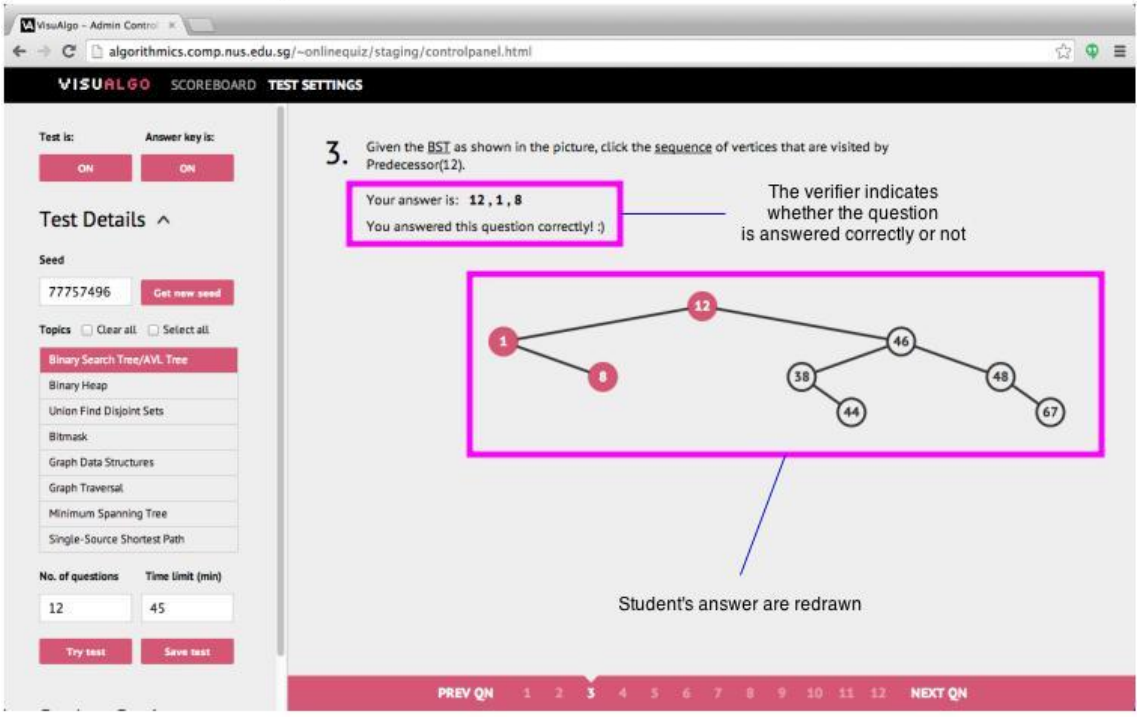
Random Graph Generator

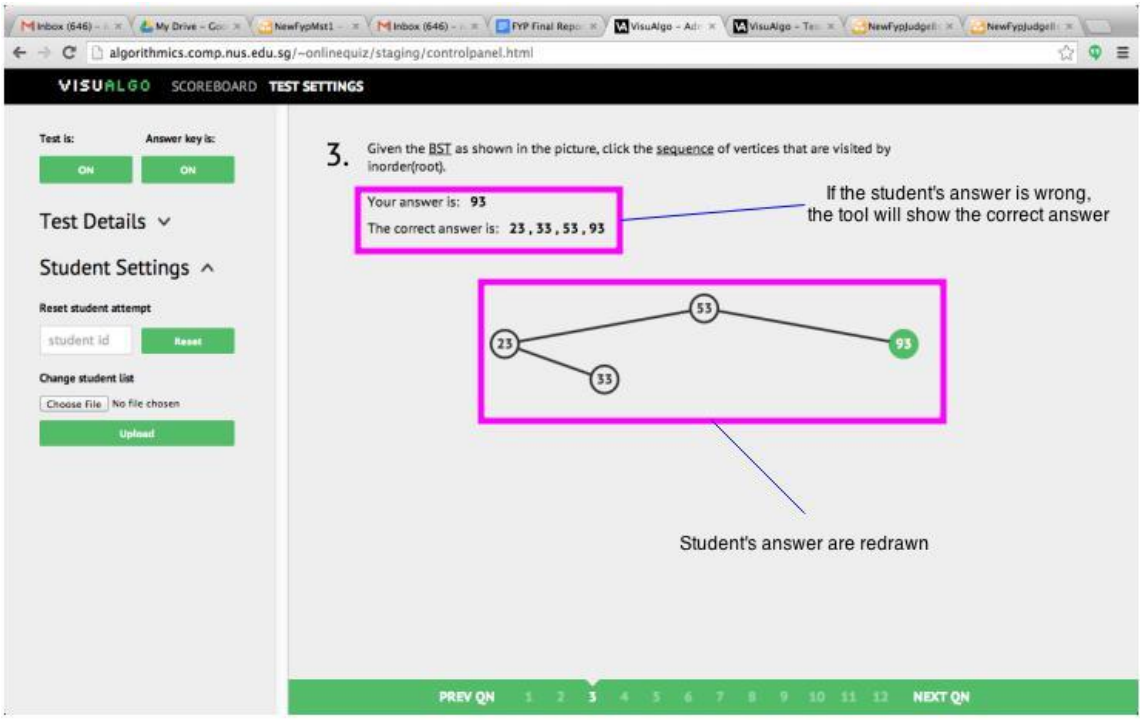


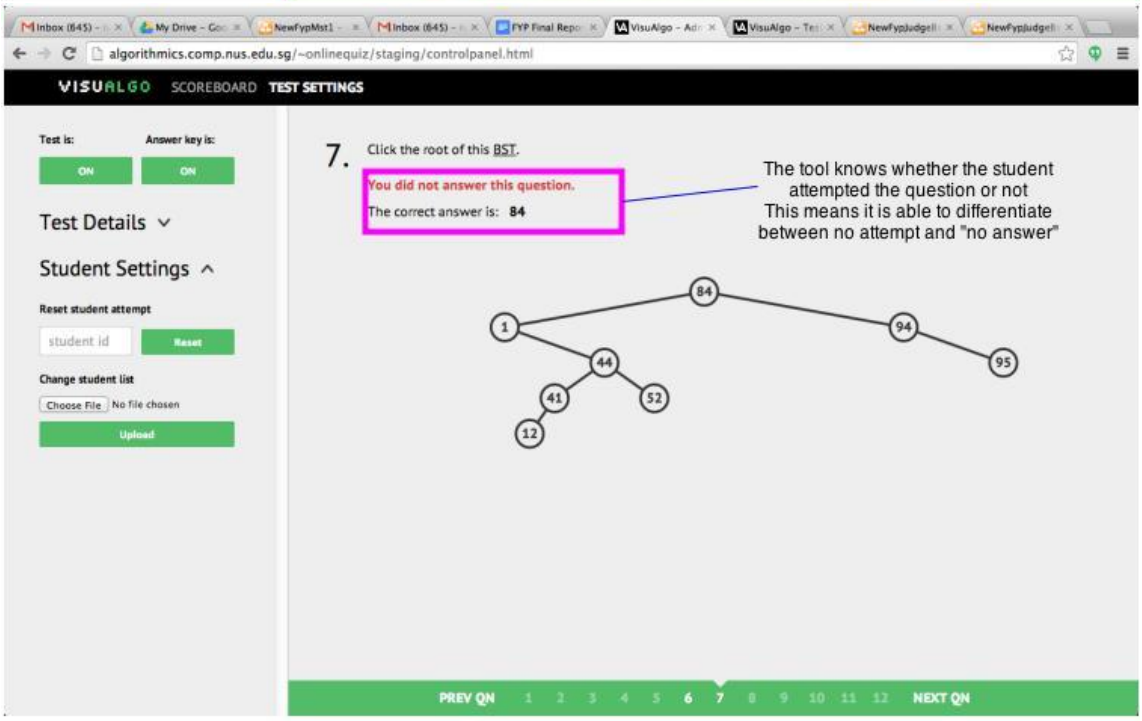


Answer Verifier

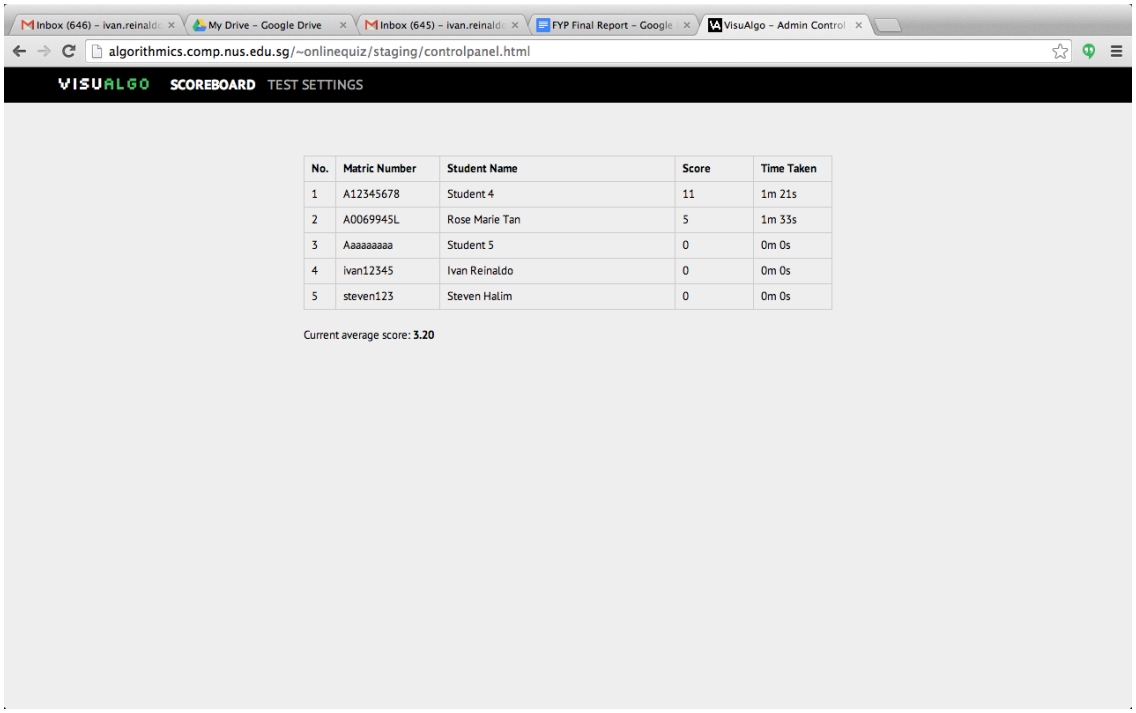





algorithmics.comp.nus.edu.sg/~onlinequiz/staging/controlpanel.html
VISUALGO SCOREBOARD TEST SETTINGS
Test is: ON
Answer key is: ON
Test Details
Student Settings
Reset student attempt
student id
Reset
Change student list
Choose File No file chosen
Upload
3. Given the BST as shown in the picture, click the sequence of vertices that are visited by inorder(root).
Your answer is: 93
The correct answer is: 23, 33, 53, 93
If the student's answer is wrong, the tool will show the correct answer
53
23
93
33
Student's answer are redrawn
PREV QN 1 2 3 4 5 6 7 8 9 10 11 12 NEXT QN


algorithmics.comp.nus.edu.sg/~onlinequiz/staging/controlpanel.html
VISUALGO SCOREBOARD TEST SETTINGS
Test is: ON
Answer key is: ON
Test Details
Student Settings
Reset student attempt
student id
Reset
Change student list
Choose File No file chosen
Upload
7. Click the root of this BST.
You did not answer this question.
The correct answer is: 84
The tool knows whether the student attempted the question or not
This means it is able to differentiate between no attempt and "no answer"
84
1
94
44
95
41
52
12
PREV QN 1 2 3 4 5 6 7 8 9 10 11 12 NEXT QN

Scoreboard


Inbox (646) – ivan.reinald
My Drive – Google Drive
Inbox (645) – ivan.reinald
FYP Final Report – Google
VisuAlgo – Admin Control
algorithmics.comp.nus.edu.sg/~onlinequiz/staging/controlpanel.html
VISUALGO
SCOREBOARD
TEST SETTINGS
No.
Matric Number
Student Name
Score
Time Taken
1
A12345678
Student 4
11
1m 21s
2
A0069945L
Rose Marie Tan
5
1m 33s
3
Aaaaaaaaa
Student 5
0
0m 0s
4
ivan12345
Ivan Reinaldo
0
0m 0s
5
steven123
Steven Halim
0
0m 0s
Current average score: 3.20

6 Recommendations for Future Works

6.1 Visualization Tool

Improve the Software Design

One of the drawbacks of the current visualization system is that it has to rely on Graph Library for visualizations, which makes it difficult for non-graph related data structures and algorithm to be developed. One solution to this is to implement another "Library", such as "Table Library" for table-based data structures and algorithms. Some works also needs to be done to make sure the "Don't Repeat Yourself" principle is held, such as by separating the animation controller from GraphWidget.js.

Graph Library itself has some features that need to be implemented. One such features is the graph drawing capabilities, which is currently have been implemented as visualization logic but too buggy to be used by students. It has been suggested that this feature is implemented as an inheritance of GraphWidget.js and thus becomes part of the Graph Library, because graph-drawing capabilities are not data structures or algorithms, but utilities to assist the visualization of graph-related data structures and algorithms.

All of these features were planned during the development, but was put on hold due to time constraint. In particular, the visualization tool is being used by the CS2010 students during the development period, and so quickly implementing visualizations used in the class takes higher precedence over system design. Moreover, the focus of the project is the online judge tool, not the visualization tool.

Add More Visualization

Currently the visualization tool has not implemented all the visualizations it needs to implement. There are still basic data structures and algorithms which needs to be implemented, ranging from CS1020 Data Structures and Algorithms 1 such as Stack and Queue to the more exotic ones from CS3230 Design and Analysis of Algorithms and CS3233 Competitive Programming such as Binary Indexed Tree and Maximum

Flow. Full list of the exotic algorithms and explanations on them can be found in Competitive Programming 3 book, in which the tool needs to implement every algorithms presented in the book (Halim and Halim, 2013). These are part of the functional requirements, and plans are also made to implement these visualizations. However, time constraint forces our team to implement CS2010 visualizations first. Furthermore, the system design has to be improved first as described in the previous section before our team can implement some of these visualizations.

6.2 Online Judge Tool

Improve the Intelligence of the Question Generator

There are several ways in which the intelligence of question generators can be improved. Two of them are listed below:

- Implement the ability to generate questions of varying difficulties
  Currently, the question generator does not recognize the difficulties of the questions; it simply generates questions randomly. As a result, some questions can be too trivial, while others can be extremely difficult to solve by humans in reasonable time. Having the ability to generate questions of varying difficulties will greatly benefit the examiners, which can then set the difficulty of the test questions more easily, and perhaps purposely generate corner-case questions. Students can also benefit by allowing them to train on varying level of difficulties according to their current level of understanding, gradually increasing the difficulty as they progress.
- Implement a more efficient answer verifier
  One of the problem with answer verifier is that there are questions that are easy to verify, but difficult to generate an answer. By difficult, I am referring to the time complexity required to arrive at the answer, since some of them seemingly requires costly brute force method. A portion of them are also NP-Hard, for example a question which requires students to draw a graph of certain size V that satisfies certain set of properties. The verifier algorithm can simply check whether the students' graph satisfies the set of properties, but it will have to generate a graph with the said set of properties to provide correct answer for

students, resulting in lots of graphs generated and verified. This is required especially on questions with "No Answer" option, because on the occasion where the students cannot arrive at a proper answer, the system is expected to generate a correct answer for them.

Enhance the Security System of the System

Security aspects have been overlooked during the developments. Compared to NUS Standard of Procedure in conducting online assessments (as stated by Dr. Halim; the original document is confidential in nature), the online judge tool is severely lacking. Some of the major security problems are listed below:

- SQL Injection
  Since security aspects are not the focus of the project, some database access classes might allow SQL injections. This will make it very easy to conduct hacking attempts, and thus should be fixed as soon as possible
- Lack of Encryption
  Passwords and students' answers are not encrypted in the database, and even worse the administrator password is hardcoded inside the system. Furthermore, it is also necessary to encrypt the network communication itself to prevent Man in the Middle attack
- System Auditing
  Students' activities have to be monitored to identify unusual actions, and that answers are saved regularly in case of network disruption / other Act of God. Currently none of these are implemented
- Secure Browser
  Actions on the browsers should be restricted, such as prevention of viewing the source code used. Currently, only minification and obfuscation of JS files are done to deter outright source code reading, but obviously this can be reverse-engineered easily even by tools searchable in Google. Further actions, such as locking browser console as described by Liew (Liew, 2014) might be necessary.

References

- Balfour, S. P. (2013). Assessing Writing in MOOCs: Automated Essay Scoring and Calibrated Peer Review, pp. 3-4.
- Delaney, J., Johnson, A., Johnson, T., Treslan, D. (2010). Student's Perception of Effective Teaching in Higher Education, pp. 33.
- Fleming, N. D. (2006). Teaching and learning styles: VARK strategies. ND Fleming.
- Galles, D. (2011). Data Structure Visualizations. Retrieved April 6, 2014, from http://www.cs.usfca.edu/~galles/visualization/
- Halim, S. and Halim, F. (2013). Competitive Programming 3 The New Lower Bound of Programming Contests. Lulu, Singapore.
- Halim, S., Koh, Z.C., Loh, B.H.V., Halim, F. (2012). Learning Algorithms with Unified and Interactive Web-Based Visualization, pp. 1-3.
- Jordan, K. (2013). MOOC completion rates: The data. Retrieved April 6, 2014, from http://www.katyjordan.com/MOOCproject.html
- Krathwohl, D. R. (2012). A Revision of Bloom's Taxonomy: An Overview, pp. 4.
- Liew, K. (2014). Disable Javascript Console in Browsers. Queness. Retrieved April 6, 2014, from http://www.queness.com/post/16151/disable-javascript-console-in-browsers
- Martin, R. C. (2000). Design principles and design patterns. Object Mentor, pp. 1-34.
- Microsoft (2009). Chapter 2: Key Principles of Software Architecture. Microsoft Developer Network. Retrieved April 6, 2014, from http://msdn.microsoft.com/en-us/library/ee658124.aspx
- Nikander, J., Korhonen, A., Seppälä, O., Karavirta, V., Silvasti, P. and Malmi, L. (2004). Visual algorithm simulation exercise system with automatic assessment: TRAKLA2. Informatics in Education-An International Journal, Vol.3, No.2, pp. 267 – 288.

- Schneier, B. (2012). Cheating in Online Classes. Schneier on Security. Retrieved April 6, 2014, from https://www.schneier.com/blog/archives/2012/06/cheating_in_onl_1.html
- The Wolfram|Alpha Team (2013). New Wolfram Problem Generator: Practice and Learn. WolframAlpha Blog. Retrieved April 6, 2014, from http://blog.wolframalpha.com/2013/10/18/new-wolfram-problem-generator-practice-and-learn/
- Ullrich, T., & Fellner, D. (2004). AlgoViz-a computer graphics algorithm visualization toolkit. In World Conference on Educational Multimedia, Hypermedia and Telecommunications (Vol. 2004, No. 1, pp. 941-948).